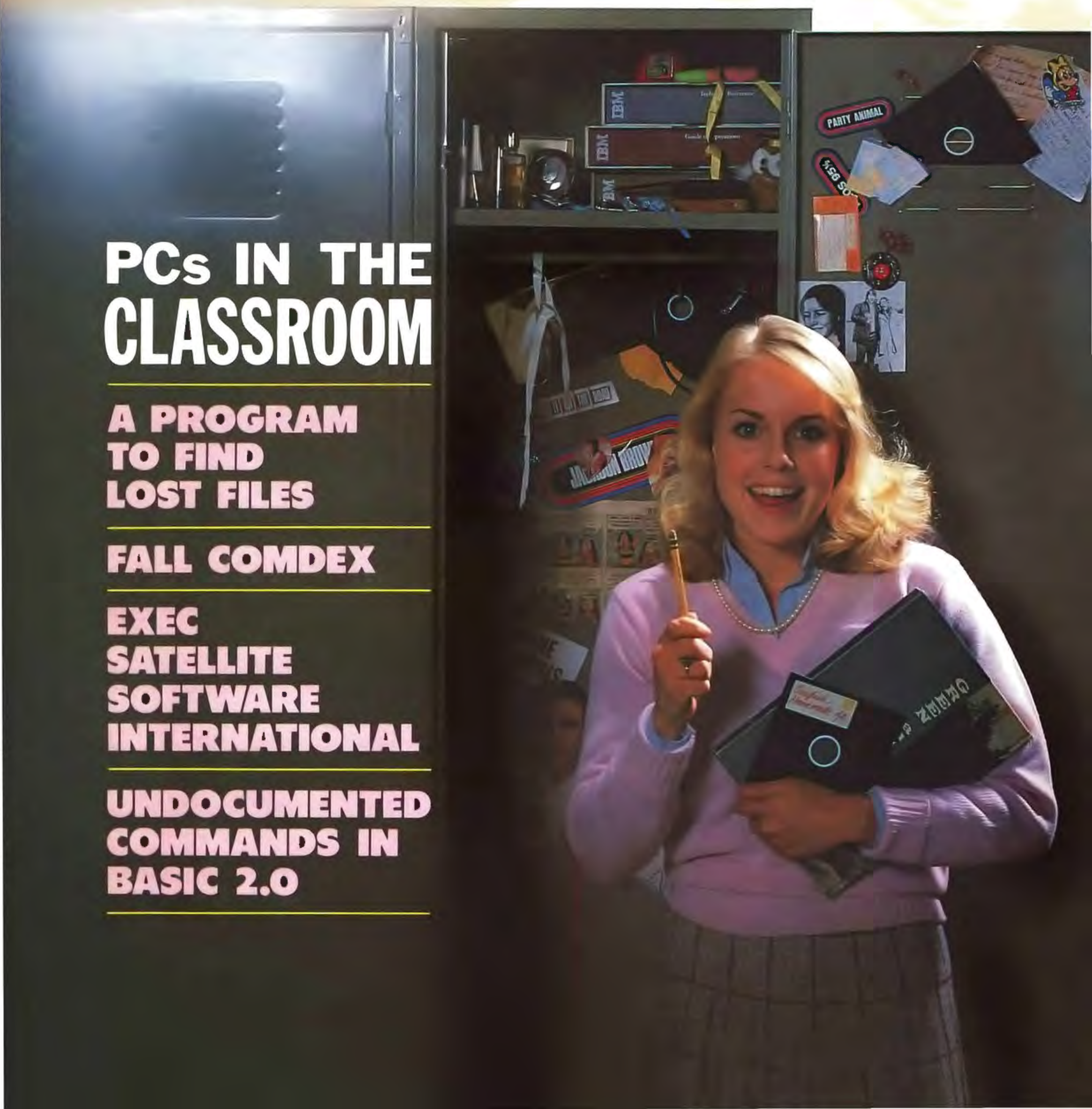
January 1984 Volume 2 $2.50

softalk

for the IBM Personal Computer



PCs IN THE CLASSROOM

A PROGRAM TO FIND LOST FILES

FALL COMDEX

EXEC SATELLITE SOFTWARE INTERNATIONAL

UNDOCUMENTED COMMANDS IN BASIC 2.0

softalk for the IBM Personal Computer January 1984

It Reads, Writes and It Paints in 3-D, Keeps and Talks to

It's called "OPEN ACCESS," and it's the result of 60 man-years of effort to create a truly do-it-all, super-program—one that can perform virtually every task you're ever likely to encounter.

The beauty of it is, all that capability resides on a single program. You don't have to re-enter data. Or spend time trying to get unmatched programs to work together.

OPEN ACCESS takes its name from the source of its power—a relational data-base manager that gives you access to more data in more ways than any comparable software.

OPEN ACCESS includes an electronic spreadsheet, 3-D graphics, word processor, appointment scheduler and telecommunications module—all revolving around the powerful information manager.

See us at COMDEX, Booth W648

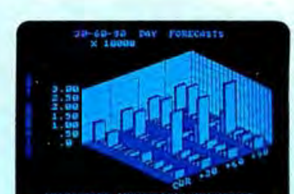
1

INFORMATION MANAGEMENT—THE MASTERMIND. This advanced data-base manager stores and retrieves multiple files quickly, easily and reliably. What's more, it shares all information with the other programs, so you never have to re-enter the same data twice.

2

ELECTRONIC SPREADSHEET—NUMBER CRUNCHING AND GOAL SEEKING. It helps you produce forecasts, cost estimates and "break-even" points—in seconds, instead of hours or days. Best of all, it allows "goal seeking." Ask, for example, "What sales must I have the rest of the year to net $1 million?," and OPEN ACCESS will figure it out!

3

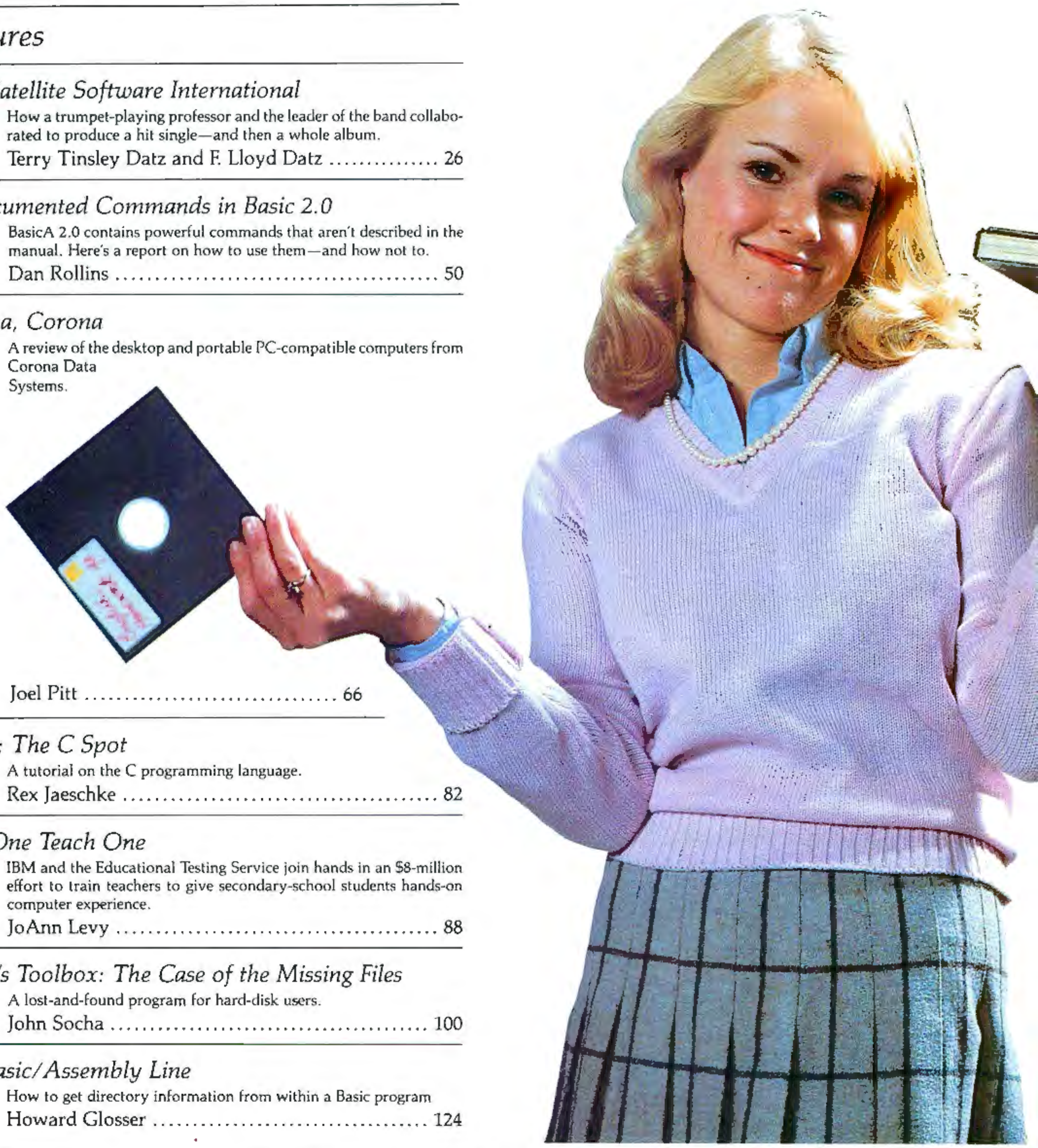


3-D GRAPHICS—NOT JUST PRETTY PICTURES. These graphics distill raw data into trends that can be instantly visualized, helping you discern the important facts from a wealth of information.

softalk

for the IBM Personal Computer

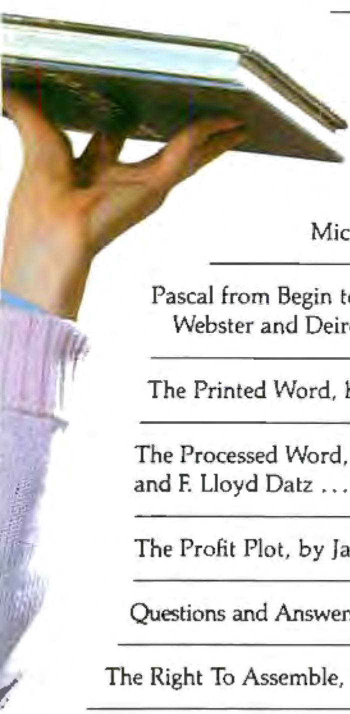
Features



Winchester Cathedral: Do You Really Need That Real Estate?

Comdex: The Rite of Autumn

Columns



Departments

Cover: Alice Cadogen's reaction to the new PCs at her school could only be described as "tubular." When asked whether PCs had a place in secondary education, her reply was, "For sure, for sure." Cover design and photography by Kevin McKeon and Tim Durr.

Index of Advertisers

softalk

for the IBM Personal Computer

Composition by Type Works, Pasadena, California. Printing by Volkmuth Printers, Saint Cloud, Minnesota.

Softalk for the IBM Personal Computer. Volume 2, Number 8.

ISSN: 0733-2173. *Softalk for the IBM Personal Computer* is published monthly by Softalk Publishing Inc., 7250 Laurel Canyon Boulevard, North Hollywood, CA 91605; telephone (818) 980-5074. Second-class postage paid at North Hollywood, California, and additional mailing offices.

Postmaster: Send address changes to Softalk/IBM, Box 7040, North Hollywood, CA 91605.

Subscriptions: Complimentary trial subscription to all owners of IBM Personal Computers or Compaq computers in the USA. If you own a pc or a Compaq but you aren't receiving *Softalk for the IBM Personal Computer*, send your serial number and mailing address to Softalk/IBM Circulation, Box 7040, North Hollywood, CA 91605. Non-pc-owner subscriptions: $24 per year. Please allow six to eight weeks for processing. *Softalk for the IBM Personal Computer* is totally independent of International Business Machines.

Back issues (from June 1982): $3.

Problems? If you haven't received your *Softalk* by the fifteenth of the month, or if you have other problems with your subscription, Marsha Stewart can help out. Call (818) 980-5074.

Moving? Send new address and a recent mailing label from your old address to Softalk/IBM Circulation, Box 7040, North Hollywood, CA 91605; telephone (818) 980-5074. Please allow six to eight weeks for processing.

MicroCommotion

Over the weekend, the offices of CompuCompany were moved to plush new digs. As usual, everything didn't go smoothly. While Ann, Carter, Ron, George, and Miriam were enjoying their Saturday, a moving company cleared out the old offices, crating up everybody's PCs and moving them.

Unfamiliar with each user's configuration, the movers made a few mistakes. Fix this mess and you win $100 worth of software.

The first mistake the movers made was to deliver each of the five PCs to the wrong place. One each ended up on loading docks in Perth, Australia; Nome, Alaska; Tokyo, Japan; Bonn, Germany; and East Cupcake, Texas. Well, at least two of them are still in the country.

Everybody's favorite peripherals and software were all mixed up and dumped into one big box. Can you straighten them out? One copy each of *WordStar*, *1-2-3*, the arcade game *Styx*, *Witness*, and *Microsoft Flight Simulator* are tangled up with a joystick, light pen, modem, memory board, and color monitor. At least all this stuff arrived in the right place.

Here's something to make your job easier. Thanks to some unique customization, it'll be easy to tell the five missing PCs apart when you locate them. One is pink, one has travel decals on it, one is plaid, one is polka-dotted, and one is candy-apple green.

What we need from you is a list of the five owners telling us whose PC is where, what color it is, and what software and peripherals go with it. Everything must be on the right desks come Monday morning.

Here are your clues:

1. Ron's PC is cooler than the rest.
2. The detective's adventure is grilling cowboys.
3. The PC in Nome can do a global search and replace.
4. Ann had the highest score.
5. George's PC would be envied in the pages of *Hot Rod* magazine.
6. Fortunately Miriam's PC has extra K rations for the long flight home.
7. His keyboard is hopping one, two, three, in some animal's pouch.
8. Godzilla wishes his hide could look as well traveled.
9. The monitor has more pixels than the system unit has dots on it.
10. Luckily, Ron could call his PC to find out where it was stuck.
11. The joystick is painted to match her pink PC.

Send your entry to Logical Moves, Softalk/IBM, Box 7040, North Hollywood, CA 91603. Please include your name, address, phone number, and what you want if you win (chosen from the products of this month's advertisers).

If we receive more than one correct entry, we'll call on the random-number generator to pick a winner. Deadline: February 15, 1984. ▲


Printer
* DOS likes to announce itself by typing A>. The cursor sits next to the >. This is called the "DOS PROMPT". When you see the DOS prompt you can count on this:
DOS IS ALIVE AND IS PATIENTLY WAITING FOR YOU TO COMMAND IT.
GIVE DOS YOUR FIRST NAME

User Memory from 64K-512K — When added to your existing system memory, brings your PC up to its maximum of 640K. You can run larger spreadsheets or create larger in-memory databases. You can also use the additional memory along with the supplied AST SuperPak™ software which includes SuperDrive™ and SuperSpool™.

Serial Ports — Using a modem your PC can communicate with other computers over telephone lines. By connecting a serial printer, you can obtain high quality print output. Other serial devices such as a plotter or mouse may also be connected to a serial port.

Parallel Ports — Used for connecting a parallel printer to your PC for high-speed, draft quality printouts.

Clock Calendar — With the on-board battery, the clock-calendar feature automatically maintains the correct date and time, even when the PC is turned off.

Game Adapter — Allows you to connect an IBM-type joystick to your PC so you can play the multitude of arcade quality games. Or you can use the joystick input for other applications by writing your own programs.

Other Communications Products — AST Research also extends the capabilities of your IBM PC with mainframe communication products including 3270 SNA and 5251 terminal emulation, 3780 RJE support, and AST-PCnet™ — the Local Area Network designed for the IBM PC.

AST *Quality*

All AST Research enhancement products come with the **AST "Plus"** — our unsurpassed reputation for quality, reliability, after-the-sale support, and overall design excellence — which give our products the best price/performance ratio in the industry!

PCnet is a registered trademark of Orchid Technology, Inc.

crosstalk

Basic *Bsave* Clarified

Your readers may be interested in a problem with Basic's *bsave* command recently confirmed by IBM. Note that IBM classifies this information as a clarification, as opposed to calling it the bug that it really is. Anyway, here's how the problem arises:

1. Cold start the PC; don't soft boot it. This must be the first operation done on the PC, which may explain why readers may never have encountered the problem with *bsave*; their machines are probably on all day, and routines requiring *bsave* may not have been the first things they did on the PC when they powered up.

2. Key in the program on page 4-33 in the Basic manual. Use *def seg = &HB000* for the monochrome screen in line 20, and use &H1000 in line 30 so as not to take up too much space on disk. Save this program to disk and go back to DOS.

3. Load Basic; then load and run the saved program. You'll probably get the following errors:

```
Syntax error in 30
Ok
Undefined line number in 30
Ok
```

4. If you pull a list at this point, you get garbage all over the screen.

That's the problem. Here's the fix:

Reload the program and run it again without leaving Basic, and it will work just fine. Also, it appears that even though the error messages are generated, the *bsave* does actually save the data it's supposed to save.

Note that this fix is for machines with more than 64K.

Howard Glosser, Medford, OR

Structured Basic

The November 1983 *Softalk* had an exceptional article by Mark Gardner ("BSCbas: Structured Programming in Basic"). However, several errors in the listing made it impossible to run the program correctly. The following are the corrections to the program listed on pages 114 and 115 (figure 6).

```
1130 IF J<8 THEN GOTO 1100
1150 IF I<=NAMECTR THEN GOTO 1070
1440 IF ERLPTR>LEN(FILLIN$) THEN
     SEARCHFLAG=0:NOWORD=1:
     GOTO 1470
1460 IF (TEST$)="A" AND TEST$<="Z")
     THEN SEARCHFLAG=0
```

It is interesting to note that the program was probably written for a machine other than a PC; it could have been simplified had Gardner chosen the single quote mark to define his comments in the source code, since IBM Basic interprets the single quote as an alternative to the *rem* statement.

Thanks for your continued good work.

Robert Harris, Stamford, CT

We're now generating typeset copy directly from program disk files. Because our programs are no longer rekeyboarded, errors like those that appeared in BSCbas should not occur in future Softalk *program listings. Thanks for keeping us on our toes.*

The Aspirin

Like many of you, I am speculating about IBM's plans to protect PC*jr* sales once it introduces its next home computer, code named the *Aspirin*.

Since the 8088 gave life so successfully to a few dinosaurs in the IBM arsenal, and since the chiclet keyboard on the PC*jr* will no doubt prolong the bullish sales of the PC, then we might as well brace ourselves for Big Blue's next mid-technology blockbuster.

For less than $500, the *Aspirin* reputedly will boast a fifty-key membrane keyboard, 2K of usable RAM, and five external expansion slots. The $995 expansion kit, nicknamed the *Jolt*, will add 16K of RAM and a choice of operating systems. Conceivably, four dealer-installed *Jolts* will allow the computer to run most PC*jr* software. To avoid an uncharacteristically Rube-Goldberg image, IBM has reportedly encouraged a variety of OEMs to develop PC-like expansion cabinets.

Come on IBM, give us the *Aspirin* now and save us all the pain of a prolonged migraine in anticipation of your next world-beater computer. The first time around, the public believed in giant GM and dependable Ford; the second time we take a closer look.

Stan Plewe, Saint George, UT

p-System Again

I feel that I must respond to C.A. Irvine's letter ("Crosstalk," November 1983), which attempted to defend the execution speed of the IBM PC version of the p-System.

Specifically, I take issue with his claim that the p-System on the PC is *not* slow. First, by design, the execution of programs under the p-System is going to be slower than that of most MS-DOS or CP/M-86 programs performing the same task. A program running under the p-System is interpreted. This means the user's program is slowed by an amount directly related to the processor overhead consumed by the interpreter. Irvine's mention of the Sage as an example of a fast p-System is absurd. The

Sage was specifically designed to run the p-System with the p-System's software deficiencies compensated for in hardware.

As the designer of the p-System turtle graphics subsystem for a personal computer built by one of the companies in the OEM list presented by Irvine, I am very familiar with the p-System. I am not condemning the p-System; it has a loyal following and it provides a level of software transportability that others can only dream about. However, it is painfully, painfully slow.

David Keenan, Austin, TX

Files in Fine Print

There is always a problem in knowing exactly what files are stored on a disk. Neither keeping a directory on a separate page in a notebook nor scribbling cryptic information on a small label is very handy. I have a method of using the printer to copy the directory information and any desired comments. The only requirements are the Epson printer with Graftrax-Plus (to use superscripts) and a magnifying glass to read the printed information.

First, it is necessary to set the printer for condensed superscripts, with a spacing between lines of 18/216-inch (or 6/72-inch). This can be done very easily using the alt key and the numeric keypad. Be sure to hold down the alt key and press each number until the complete number is entered. Type:

COPY CON: LPT1:
ALT-155 S ALT-128 CONTROL-0 ALT-155 3 ALT-18
CONTROL-Z

The printer is now set for very small printing. When you type control-print-screen, the printer will keep a log of every command for you. Thus, you can type *dir/w*, and a wide directory will be listed. If you have sorted a directory program called SD, then typing *SD* will produce the output on the paper. Any time you wish to enter a comment before the directory listing (perhaps to describe an entire disk) just use "this is a comment line"—starting in column 1.

Remember to type control-print-screen again to cancel the printing. To reset the printer to its power-up state, type

COPY CON: LPT1:
ALT-155 @
CONTROL-Z

Judith R. Epstein, Highland Park, IL

***WordStar* in Color**

I have had a lot of people ask me, "How did you do that?" when they see color on my *WordStar* 3.24. It's actually very easy to modify the IVON and IVOFF parameters of *WordStar*, using the DOS program Debug, to modify the video attributes.

For starters, copy Ws.com onto a scratch disk, just in case you make a mistake, and then copy Debug.com from DOS. The underscored words are what you type.

A⟩DEBUG WS.COM	
–	DEBUG prompt
–e 284	Location of IVON
0905:0284 xx.13	xx is the present video attribute, 13 is blue background with cyan letters.
–e 28b 0c	This changes IVOFF to cyan letters.
–w	Write what you did to
–writing 5180 bytes	disk.
–q	Leave debug.
A⟩	Done!

Now you can run *WordStar* and see what it looks like. The attribute bytes vary from 00 to 0F in hex for color letters and 10 to 1F for inverse video. On monochrome displays the response is different. I didn't bother with predicting the result and just tried all the values until I saw what I liked.

If you make a mistake, particularly in the

continued on page 13, column 2

SOFTALK CLASSIFIED ADVERTISING

Adventure

MAP OF COLOSSAL CAVE
For IBM PC and other versions of "ORIGINAL" ADVENTURE GAME. A detailed map of the COLOSSAL CAVE including travel instructions, treasure locations and magic words. $6 Postpaid. BLUEJACKET SOFTWARE, Dept. ST, Box 13547, St. Petersburg, FL 33733.

Business

STATISTICIAN'S MACE
Easy to use program calculates descriptive stats, multiple regression, correlations, several ANOVAs, nonparametric tests, and other statistics used by scientists, business researchers, engineers. Accepts keyboard or disk file input. Requires IBM PC, 128K, IBM PC or MS DOS, 1 disk. Price $195. Evaluation version $30. MACE Inc., 2313 Center Ave., Madison, WI 53704; (608) 241-4566.

TAX83/PC
Use with your spreadsheet program for 1983 taxes. Computes interactively with Form 1040, Schedules A, B, D and W. Also 1984 estimated taxes. Specify Lotus 1-2-3, Multiplan, VisiCalc, PeachCalc or SuperCalc. On disk by first class mail. $14.95. Tax Assistance Associates, 3410 Lodge, Belmont, CA 94002.

DATA SECURITY—$34.95
Secure sensitive data files with FAST-CRYPT. FAST-CRYPT is an efficient Assembly language implementation of the powerful DES (Data Encryption Standard) algorithm. Encrypt or decrypt any disk file at 120,000 bytes/minute. Unprotected disk w/manual-$34.95. IBM-PC, XT & compatibles 64K PC/MS-DOS. KIWI Software Co., Box 218 S, Plainsboro, NJ 08536.

SMALL FORTUNE AUTOLEDGER
A GL for small business—Simple & efficient
* Single entry or self-balancing double entry
* Powerful—400 GL accounts & 8 departments
* Not Protected—use with RAM & hard disks
* Fully custom reports, *even basic A/R, A/P*
* Binder/slipcase incl tutorial, acct. theory
* $250, 96K req'd (200 GL acct's, 64K, $200)

Visa, Mastercard, COD Accepted
EMERALD Software, Inc.
(206) 282-2100, Box 19453, Seattle, WA 98109

1099'S MADE EASY
Stop IRS penalties—CPA designed. Fill in form on screen and print. Files reusable next year.
Prepares 1099 MISC / 1099 INT.
Requires dBASE II, 80-Column, two drives.
Single payor $75, Multi payor $90, + 6% tax.
SHIRTSLEEVES SOFTWARE, 3300 Irvine Ave., #265,
Newport Beach, CA 92660.

1040PLAN TEMPLATE $45
1983 Tax planning & preparation template for 2nd generation spreadsheet Lotus 1-2-3 & SuperCalc III. Flexible, up to four different alternatives can be computed at once. Follows IRS forms, by line number, includes: form 1040, schedules A,B,C, D,E,G,W, & SE & forms 2119, 2441, 3468 & 6251. Req. 256K.
William A. Permar, CPA, 1125 Sunnyhills Rd.
Dpt. ST, Oakland, CA 94610
(800) 227-1617; In CA (800) 772-3545 exten. 644.

POINT OF SALE
PC CASH REGISTER turns your PC into a cash register. FEATURES: inventory, sales reports, credit controls, commissions and customer base. OPTIONS: cash drawer, purchase orders, mail labels and invoicing. $295 & $395. Write or call (415) 927-0990. Data Consulting Group - DATA*EASY Software, 12 Skylark Dr. #18, Larkspur, CA 94939.

Graphics

THE PROWRITER UTILITIES
Complete control of your C.Itoh 8510/1550 and NEC 8023. DOS 1.1 & 2.0 compatible for $44.95.
PROCNTRL- On-Line Keyboard Control of printer.
PROEPSIM- Epson Graftrax Simulator (eg 1-2-3).
PROSCR- Text/Graphics Screen Dump, 2 sizes.
PROSTALL- WordStar 3.2 & 3.3 Installation.
PROSET- Menu Driven Printer Setup & 2 more.
COURTRIN ENTERPRISES, Box 231190
San Diego, CA 92123; (619) 569-8308

PC ART DISK AND BOOK
Interested in using your PC for a non-figurative geometric artistic experience? Req. Color Bd. & Mon. Try one of many by noted sculptor & educator Leroy Lamis. $50.

```
10 screen 1: key off: cls: color 4,1
20 for X = 5 to 100
30 circle (160,100),215,rnd*2+1,,,5/X
40 circle (160,100),215,rnd*2+1,,,X/5:next
```

PC ART, 3101 Oak St., Terre Haute, IN 47803.

PC-TITLE/PC-PROJECTOR
Create full color, professional-quality text for slides, screen displays and flyers. Produce manual- or time-controlled "slide shows" using the IBM PC as an electronic slide projector. Add full graphics with PCcrayon. PC-Title/Projector—$49.95. PC-crayon—$44.95. Mention this ad and get all 3 programs for $79.90. VISA/MC. PC Resources, Inc., (408) 243-4169.

Home

TOTAL FITNESS PROGRAMS
Your computer can help you & your loved ones live longer & enjoy life more. Send for "free" price list of Fitness Programs including nutrition, exercise, psychology & educational health games.
CTRL-HEALTH SOFTWARE, Dept. B
18653 Ventura Blvd., #348, Tarzana, CA 91356.

Services

BORED IN '84??
RENT today's most popular computer software for pennies a day! FREE brochure.
The Soft Source-R Inc.
Dept. K, Box 2931, Joliet, IL 60434

SAVE AT GOLEM COMPUTERS
Our ** SOFTWARE and HARDWARE** prices are lowest. All major brands are available. We carry business, education and entertainment software. Call for **FREE** catalog. (800) 345-8112, in Pennsylvania (800) 662-2444.

SOFTWARE SPECIALS
Call Name-Brand Software for the best prices and fast delivery. Specializing in business software. . . .Visi-Series, PFS: Series, Microsoft, etc. Call toll free: (800) 356-7511. In Wisconsin, call (608) 754-7527. Visa and MasterCard accepted at no extra charge.

SUPER SOFTWARE SAVINGS
For a complete catalog of personal and small business computer software and hardware at excellent prices, write: SBCC, Box 1191, Thousand Oaks, CA 91360 or phone (805) 492-9391.
Service is our Motto!

3M SCOTCH DISKETTES $20.95
Authorized 3M distributor. Buy wholesale. 5.25" SS/DD $20.95. DS/DD $29.95. Reinforced hub. Why buy generic? Complete price list available. Call (415) 778-2595 or write Argonaut Distributing, 1104 Buchanan Rd. STI, Antioch, CA 94509. Prompt delivery!

Utility

END PRINTER FRUSTRATION!
SurePrint takes the pain out of using your printer! Printer options made easy for any IBM PC package, from DOS, BASIC, and on-line. SurePrint available for IBM (NEW COLOR, TOO), all Epson, all Okidata, IDS, C.Itoh, NEC, Anadex, TI, Juki, TEC, Diablo, others. Only $35 for one printer, add'l printers $15. Specify make and model. NYS add 7 1/4% tax.
Dickinson Associates Inc.
Box 1358, Melville, NY 11747

TIMEPIECE BY TECHART
THE $3000 PERSONAL CLOCK
Turn your IBM PC into a desktop clock. Up-to-the-second digital and analog report of current time, plus elapsed time, countdown and the unique message alarm functions. Luminous dial and sweep second hand. Ideal gift for the person who knows the value of time. Requires graphics monitor. Send check or MO for $19.95 to: TechArt Associates, Box 370, Berkeley, CA 94701; (415) 525-0870.

addresses for IVON and IVOFF, the results are unpredictable, hence the need to work with a copy of Ws.com. If you do make a mistake, just start over. If you want more information on how to patch *WordStar*, you might want to take a look at Appendix C of the *WordStar* manual. This contains an assembly listing of the user patch areas and the location of IVON, IVOFF, and other user patches. Another good reference for using Debug to patch *WordStar* is Emil Flock's "WordStar Made to Order" in *PC World*, Vol. 1, No. 2, which gives the addresses of most of the patches in a convenient reference.

Robert A. Day, Livermore, CA

Artwork on Screen

For your reader who requested a more precise definition of RGB monitor colors to create artwork or screens, I have provided the following list of colors.

Color	*IBM Basic No.*	*Pantone Colors*
Black	0	Process Black C
Blue	1	Process Blue C
Green	2	368 C
Cyan	3	305 C
Red	4	192 C
Magenta	5	Purple C
Brown	6	153 C
White	7	Process White
Gray	8	428 C
Light Blue	9	Process Cyan C
Light Green	10	366 C
Light Cyan	11	304 C
Light Red	12	189 C
Light Magenta	13	244 C
Yellow	14	121 C
High-Intensity White	15	White Process C

Pantone is a color standards company, and construction papers to their standards can be purchased at most drafting supply stores.

Clynton Yarter, Orange, CA

Escaping with *Volkswriter*

After reading John Dickinson's November "The Printed Word," I thought I'd bring a well-known piece of software to your attention that does allow direct entry of the escape character: *Volkswriter*. Merely pressing the escape key while editing will place the escape character on the screen at the cursor position, where it will be sent to the printer at print time.

The main thing to remember is to insert the escape character and its associated control characters after the text has been formatted on the screen and the margins on the tab line have been relaxed. This will prevent the addition of characters from scrambling the screen because these characters are nonprinting but still take up character positions on the screen. That is exactly the method used to produce boldface and underlining.

Volkswriter can be similarly used directly to write batch files containing the escape character, such as those used for extended screen control under DOS 2.0. In fact, most ASCII codes below 32 can be entered directly into text, and I believe that any of them can be entered following a "verb(atim)" command on the left side of the screen.

Robert Relf, Bothell, WA

DOS File Size

While converting *Mr. Lister* (the mailing list program developed and marketed by International Computers) to run using PC-DOS 1.0, 1.1, and 2.0, the problem of maximum file size came up. Using the calculations below I have come to the conclusion that *Mr. Lister* version 2.0 will support up to ten million mailing list entries sequentially indexed by company name, contact name, and Zip Code.

How large can a file be under DOS 1.0, 1.1, or 2.0? I am not sure, but here is an educated guess. Page E-12 of the DOS 2.0 manual specifies that bytes 16 through 19 of the file control block contain file size in bytes. On page C-6 it specifies that bytes 28 through 31 of each record in the disk directory contain the corresponding file size. In both cases this means DOS 2.0 uses two words to store file size. This seems to indicate that files may be as large as hex FFFFFFFF bytes in length. On page C-8 the manual specifies that absolute cluster number is a three-byte value. This means that files may not be larger than hex FFF, or decimal 4,095, multiplied by the size of clusters.

Actually, files would be smaller than this, since cluster numbers 0 and FF0 through FFF are used for special purposes. Page B-6 states that clusters are one sector, or 512 bytes, for single-sided disks and two sectors, or 1,024 bytes, for double-sided disks. This implies that if all clusters were allocated to a single file, the size of that file would be 4,095 times 512 bytes (approximately two megabytes) on single-sided disks, and 4,095 times 1,024 bytes (approximately four megabytes) on double-sided disks. Since disks are just not that big, this size limit seems acceptable.

The limits on hard disk size are of more interest. Page B-6 states that cluster size of fixed disks is determined at format time and is based on the size of the DOS partition. If cluster sizes are large enough, it seems reasonable that file sizes can be as large as you wish. Pages C-6 and E-12 indicate the only restriction; that is, two words may not store a value larger than FFFFFFFF. Hence I assume files on a hard disk drive could be as large as FFFFFFFF bytes (truly a very large file).

Any comments on my observations? Does anybody have information from Microsoft about file-size limits?

Elmer W. Fittery, Mexico Beach, FL ▲

QUESTIONS & ANSWERS

by Nancy Andrews

Q: In the February 1983 issue you advised Steve Smith that several problems in the Cobol compiler had been corrected by a "just released" version 2.0.

I am interested in acquiring the compiler, but none of the vendors I contacted is aware of version 2.0. The IBM Product Center is offering an updated version 1.0. Could that be the release you referred to? If so, does it accommodate files spanning more than one disk? Otherwise, how does one acquire the corrected version?

Ward L. Darby

A: Microsoft has a new Cobol release, version 1.07; it corrects some of the earlier problems, but it has not been released by IBM. You should be able to get it from your dealer or directly from Microsoft. It does not, however, accommodate data files that span more that one disk; our earlier report was incorrect. In fact, we know of no microcomputer Cobol with this capability. Your best bet would be a hard disk.

Q: I am having some problems with my IBM *Professional Editor.* Would you consider the following two problems? IBM has already had a go at them and hasn't resolved them.

Problem 1: I am using *Professional Editor* on my pc with a green-phosphor NEC monitor. When I enter the editor, the screen comes on in reverse video, but it can be changed to the black/white screen option in the editor menu. However, when I exit the editor, the screen default mode is again set to reverse video. With existing utilities I am required to reboot DOS every time I leave the editor, just to correct the screen mode. I find this condition unacceptable and hope I might have missed some helpful documentation. Is there an internal software solution to this problem?

Problem 2: I have written a Pascal routine that reads a line of a text file in to an eighty-character string variable that is in turn written to the printer. The routine prints a DOS file in page lengths with headings, and so on. The problem is that the spaces in the text file created by the tab key by way of *Professional Editor* are read as single spaces. The proper tab spaces are read, however, by the DOS *type* command. My final printout is left-justified except for spaces created by the space bar. Can this problem be identified and corrected? IBM says the tab key creates only single spaces in ASCII code.

Idio Video and Tabs in Shabs

A: We, too, are trying to get an answer from IBM for your problems and so far have had as much success as you have. If something comes through, we'll let you know. In the meantime, to reset the screen mode you could use the Clear program listed in the October 1983 "Right To Assemble" column. It's only 109 bytes long, so it will probably fit right on your editor disk. If you run it without specifying parameters, it will clear the display and reset the normal mode—white on black.

Your tab problem is also a problem with the editor and not with Pascal. When using either *WordStar* or *Edlin* to create input files for printing with a Pascal program, we had no problems getting the tabs to print correctly. Since we've been unable to get any response from IBM or the author of *Professional Editor* (IBM's policy actually prohibits their software authors from supporting their products) the best we can do is suggest that you use a product like *ProKey* or *Keynote* to redefine the tab key or one of the function keys to five or eight blanks (ASCII 32s). We would welcome a solution from a reader who has solved this problem with the *Professional Editor.*

Q: I am seeking a general-purpose graphics program for displaying engineering and scientific data on a pc. The necessary features would be the ability to draw multiple curves on one set of axes, log-log axes, two-dimensional contour plots, three-dimensional perspective surfaces, map projections, and so on. Do you know if such a program exists?

William R. Goodin, Ph.D.

A: The programs you should look at are *MicroCAD* (Computer Aided Design, 764 Twenty-fourth Avenue, San Francisco, CA 94121), *AutoCAD* (Autodesk, 150 Shoreline Highway, Mill Valley, CA 94941), and *The Drawing Processor* (BG Graphics, 824 Stetson Avenue, Kent, WA 98031).

Q: I wish to combine four lists of names into a single alphabetized list. Each list consists of one thousand to two thousand names, and each list is already alphabetized. Thus, simply combining the four lists and doing a sorting operation seems like a brute force approach. Your suggestions for a more elegant method would be appreciated.

Howard A. Chin

A: What you want to do is a merge sort. You can set up a four-way merge sort by making each list a separate file and opening an additional file for sorted output. Then, write a program that

FRIENDLYWARE™
PC INTRODUCTORY SET
MASTER DISKETTE No.1

1) History and introduction to Computers
2) Operating Systems
3) Languages
4) Screen Prompts
5) Memory and Storage
6) Files, Fields, Records
7) Helpful DOS Commands
8) Anatomy of a Program
9) Mastermind
10) Nevada Dice
11) Killer Maze
12) Sea Battle
13) Boggy Marsh
14) Hangman
15) Tic Tac Toe
16) Home Vision
17) Home Hearing

FRIENDLYWARE™
PC INTRODUCTORY SET
MASTER DISKETTE No.2

1) Wildcatter
2) You Draw It
3) Peg Leap
4) Dominoes
5) Match
6) Towers of Atlantis
7) Personal Biorhythms
8) PC Golf
9) Head Coach
10) Blackjack
11) Othello
12) Biorhythm Sports Predicting

FRIENDLYWARE™
PC INTRODUCTORY SET
MASTER DISKETTE No.3

1) Check Book Register
2) Amortization Analysis
   - Calc. Interest Rate
   - Calc. Payoff Amount
   - Calc. Loan Amount
   - Calc. Loan Payoff
3) Inv. Reorder Anal.
4) Present/Future Value
5) Break Even Analysis
6) Economic Order Analysis
7) Depreciation Analysis
   - Straight Line
   - Double Declining
   - Units/Production
   - Sum of Years Digits
8) Computer Accounting Demonstration
9) Stock Ratio Analysis
   - Current Ratio
   - Quick Ratio
   - Inventory Util.
   - Inventory Turnover
   - Fixed Asset Util.
   - Total Asset Util.
   - Debt-to-Worth
   - Operating Ratio
   - Profits-to-Worth
   - N. Profit/N. Sales
   - ROI in Assets
   - Price to Earnings
   - Earnings per Square

THE 3 DISKETTE CURE FOR COMPUTER-PHOBIA

The novice computer user wants more than a lesson.

He wants an easy and enjoyable way to get to know his machine, feel at home around it, and find out what he can do with it.

Tens of thousands of PC and XT owners have gotten what they wanted with the **Intro Set.**

It's proving to be the perfect blend of education, entertainment, and practicality for the new user.

Nothing else on the market has helped more people overcome "computer phobia" than the **Intro Set.**

Suggested Retail: **$49.95**

Compatible with the IBM PC or XT — DOS 1.1 or 2.0

FRIENDLYSOFT, INC.

ARLINGTON, TEXAS

Producers of innovative, quality software products EXCLUSIVELY for IBM Personal Computer.

looks at the first item in each of the four input files, compares them, and writes the one closest to the beginning of the alphabet to the output file. Now look at the next item in the file you just wrote from, compare it to the original three words, and, again, write the smallest to your sorted file—and continue in this way.

A four-way merge sort obviously works only if you have four lists to sort. This is more elegant than brute force, but possibly it would be even more elegant if you wrote a two-way merge sort. That way you could use the same program over and over to sort any number of lists.

Actually, it might be most efficient in terms of your time to use the brute force method. Just start the program, go have lunch, and chances are that when you return it will be finished. This might be better than spending lunch writing the merge sort.

Q: I was most interested in your answer to Larry Ellis (September 1983) concerning the Basic stack and where it resides. I have faced a similar problem, and, although your answer to Mr. Ellis helps, my problem goes further.

I increased my PC from 64K to 128K expecting to get a 64K space of contiguous, usable memory for a program I'm working on. The second 64K didn't turn out to be adequate because the stack was relocated into it, so I bought a third memory extension. Now that I have 192K in the machine, there is a contiguous 64K zone available, and I'm continuing with my program.

But I have found that there is more residing in segments &H1000 and &H2000 than the stack. It would be very helpful to me to know what's out there and what it does.

Here's what I did:

First I brought up DOS 1.1 from a cold start to ensure that all unused memory was loaded with zeros.

Then I loaded BasicA and ran a program that locates programs' areas in memory and reports the beginning and ending addresses for each. I ran it once with *def seg = &H1000* and once with *def seg = &H2000*. It reported the following areas in use:

&H1000:326F to 328D
&H1000:6C1B to 74AC
&H2000:EF43 to EF44
&H2000:EF8C to EF9D
&H2000:F040 to FFFF

Next, I turned off my machine and restarted it with DOS. This time I loaded Debug rather than BasicA and looked at the memory contents of the addresses given above. Here's what I found:

&H1000:6C1B through 74AC was no longer in use, indicating that it was part of BasicA.

Similarly, &H2000:EF8C through EF9D was now *blank*.

Judging from what I saw in memory, &H2000:F040 through FFFF is likely to have something to do with the BIOS. &H2000:EF43, a single character, contained the value &H30.

Can you help me figure out what these things are and how I can safely make use of the memory around them (or even better, move them to a more convenient location)?

Jan Young

A: You've done a good job of sleuthing and have essentially discovered the three parts of Command.com. &H1000:326F to 328D contains Command.com's initialization code and stack. This area is wiped out by the first program run after booting but is not zeroed by Basic. The second part of Command.com begins at &H2000:EF43 and continues through EF8C. This is Command.com's data area, and it apparently has a block of zeros in the middle. &H2000:F040 to FFFF contains Command.com's transient code, the code that runs batch files and executes interrupts. You correctly defined addresses &H1000:6C1B to 74AC as belonging to BasicA; they are BasicA's stack.

As you've discovered, you have well over 64K for your program between Basic's stack and the Command.com data area. Unfortunately, you can't relocate the parts of Command.com to make a larger contiguous block of free memory.

Q: I own an IBM PC and use *WordStar* for my word processing needs. I recently acquired a copy of DOS 2.0 and quickly set about using the tree-structured directories to my advantage. It was most useful with *WordStar* because it let me group related files for easier manipulation. The addition of the *MailMerge* and *SpellStar* options to my system gave me an extremely versatile word processing station.

You can imagine my dismay when I found that *SpellStar* would not function properly. Although it would check the file for spelling errors and list them, when I tried to flag the mistakes in the text I would receive the *WordStar* message "ERROR E2—ERROR MESSAGE ERROR * * * HIT ESCAPE KEY." This occurred even when I returned to *WordStar* after aborting *SpellStar*. There have been times when it would be checking a file and about halfway through it would crash, leaving the disk spinning uselessly.

I have tried *SpellStar* with DOS 1.1, and it functioned perfectly. Would my problems be caused by DOS 2.0? If so, I am reluctant to give up the benefit of the tree directories. Knowing MicroPro's reputation for helping customers, I decided to put my questions to you first. I am a steady reader of *Softalk* and find your magazine very useful and informative. You have solved other problems and I hope you can solve mine.

Edward Owen

A: You're right. Your problem is caused by using both *SpellStar* and DOS 2.0. It took a while, but we were able to get through

Teach yourself software in minutes. Not hours.

Insert an ATI training disk, and presto! Three minutes later, you're practicing usable skills. Within an hour, you're proficient.

Even if you're never touched a computer before.

ATI's self-prompting disk simulates the actual software on the screen of your personal computer. If you have a question later, the helpful handbook is set up to give you the answer in 15 seconds.

ATI Training Power is faster because it's Performance Based. Instead of taking time teaching you how a program works, ATI shows you how to put it to work. Right away.

ATI's unique Performance-Based Training is widely used by many Fortune 500 companies, such as IBM, DEC, Xerox, NEC, and Toshiba. Not to mention thousands of small businesses.

ATI's training programs operate in virtually every business oriented microcomputer.

Match ATI's selections with your software library, and order today.

Integrated Software

ATI Training Power makes it easy to learn and use the powerful new integrated software packages currently entering the market.

Word Processing

ATI's Word Processing Series teaches you how to create, edit, move, and print text with popular programs, fast! A must for executives and secretaries alike.

Database Management

To get the most out of your data base, learn to use these programs fast, with ATI's Database Management Series.

Financial Planning

What if you could learn to do a complete "what if" spreadsheet analysis in less than an hour? ATI's Financial Planning Series teaches you this and more. Turn software packages into powerful management tools, quickly.

Operating Systems

Learn to prepare, monitor, maintain, and troubleshoot your own system like a pro, with ATI's Operating Systems Series.

Programming

ATI's Basic Programming Series teaches you how to write, enter, debug, and use programs that analyse business expenses, organize mailing lists, and more.

Accounting

Now it's easy to put your accounting on computer, with an outstanding software package and ATI Training Power to get you (and your bookkeeper) up to speed, rapidly and cost-effectively.

Please rush me ATI Training Power programs, at $75 each for this software:

Integrated Software
- ☐ Lotus 1-2-3

Database Management
- ☐ dBASE II-vol. 1 & 2
- ☐ EasyFiler

Word Processing
- ☐ WordStar-vol. 1 & 2
- ☐ EasyWriter II
- ☐ Perfect Writer
- ☐ Spellbinder
- ☐ Benchmark

Financial Planning
- ☐ Multiplan
- ☐ SuperCalc
- ☐ VisiCalc
- ☐ Perfect Calc
- ☐ EasyPlanner
- ☐ Microplan

Operating Systems
- ☐ PC DOS
- ☐ MS DOS
- ☐ CP/M

Programming
- ☐ BASIC
- ☐ APPLESOFT BASIC

Accounting
- ☐ BPI Gen. Acct.
- ☐ Peachtree General Ledger*
- ☐ Peachtree Accounts Receivable*
- ☐ Peachtree Accounts Payable*

*PC(MS) DOS formats only.

My disk size is:
- ☐ 5-1/4"
- ☐ 8"

My computer brand is: ____________________

Enclosed is $75 each plus $2.50 shipping. (CA add 6.5% tax)

Name ____________________ Phone (____)

Street ____________________

City ____________ State ____________ Zip ________

Mail order form to:
Software Training Company,
A Division of ATI
3770 Highland Avenue, Suite 202B
Manhattan Beach, CA 90266
Telex: 364412 INTR

Dealers:
call for sales kit
(213) 546-4725

For rush order call:
(213) 546-5579

VISA MasterCard

ATI Money Back Guarantee
If you're not completely satisfied with how fast you learn with ATI Interactive Training Programs, simply return it within 3 days for full purchase refund.

My computer uses:
- ☐ PC DOS
- ☐ MS DOS
- ☐ CP/M
- ☐ CP/M-86
- ☐ XENIX
- ☐ APPLE II CP/M
- ☐ APPLE IIe 80 col.

ATI TRAINING POWER™

to MicroPro, and they'll be sending you a patch to correct the problem. They mentioned that anyone else who uses DOS 2.0 with *WordStar 3.2* and *SpellStar* can write them and obtain the patch. Their address is:

MicroPro International Corporation
33 San Pablo Aveune
San Rafael, CA 94903

These problems are corrected in *WordStar's* latest upgrade—version 3.3

Q: Last Christmas I purchased a Tandon two-sided drive as a second drive for my pc. This drive worked fine for about six months. Then one day I turned my system on and began to get a series of read errors. It soon became apparent that the system now believed my B drive was single-sided. Disks that are formatted on this drive format to 160K, and when I run the diagnostics it asks if B is a 160K drive.

Recently I added additional memory to the system and suddenly the drive corrected itself back to 320K. After two days, however, it relapsed into being a 160K drive. I've tried cleaning it, but that doesn't solve the problem. All the connections seem okay. I would appreciate any suggestions you may offer to solve the problem.

Bruce Nissim

A: We contacted several technicians in our area to see what they would recommend, and the consensus was that the problem was the drive itself, not the controller, and that it was most likely a loose chip on the drive. One technician said he would just replace the drive—that it was too time-consuming, complicated, and costly to diagnose and repair them. Another said he would look at it and repair it for about $120. Yet another said he would look at it and, if possible, repair it for $75 plus parts. So it looks like it has to go to the shop, but you'd be well advised to do some comparison shopping first.

Q: Your discussion in the September "Questions and Answers" column about Basic's stack overrunning a program prompted me to write about a problem I have been unable to solve.

I wrote a program to sort literature references by keyword or author. It has an overlay menu that loads the file of references and then calls various subroutines to search, add to, delete from, modify, or print the file. The program works fine in interpreted Basic, but when I compile it I get a strange bug. The subroutine that loads the file also counts the records and passes the count in a variable, Recsum, to the other subroutines. If I have the program print out Recsum at several points, I find that when I leave one of these routines to return to the menu, Recsum gets reset to zero. However, if I return to the subroutine I just left or to any other one, Recsum now has the proper value. Furthermore, if, in the subroutine that adds records, Recsum is increased, not only is Recsum set to zero on return to the menu, but also if I then return to any subroutine without first reloading the whole file (and recomputing Recsum), the old value of Recsum is returned—not the new larger one. It almost appears as if compiled Basic is storing the variable in two different locations that are not updated simultaneously.

Do compiled and interpreted Basic pass variables differently between subroutines?

John J. Houser

A: Without seeing a listing of your program, it is hard to determine the exact cause of your problem. However, if you are using the Basic *chain* statement to join the various parts of your program, be sure you have included a *common* statement naming the variables you want passed from program to program. If you do not include this, all variables are lost; this could account for the disparate values of the variable Recsum.

Q: I just got a Vista multifunction card, which has a parallel port. The original parallel port is on the printer adapter card. I have two parallel printers, one an Epson and the other an Olympia typewriter. I configured the Vista card for Lpt2:, according to the manufacturer's instructions. The problem: I can't get the typewriter, which is hooked into the Vista card, to work. However, it works fine when hooked into the Lpt1: port. I wrote a command recommended by my dealer into the Autoexec.bat file that reads:

```
COPY CON SECOND.BAT
MODE LPT2:80,6
```

but when I run this nothing happens. Can both printers be used, and, if so, what is the proper command? If you have a solution I would appreciate your informing me in simple language since I'm a novice.

David Altman

A: Unfortunately, you can't use the *mode* command to redirect output from one parallel printer to another. Fortunately, however, the April 1983 issue of *Softalk* includes a program called Printsw, by John Socha, that accomplishes this redirection for you. Each time you want to change from one printer to the other, just run this program and all should be well. The only problem you might encounter has to do with a print spooling program; if you use a spooling program, check your documentation to see what provisions the spooler has for changing between Lpt1 and Lpt2.

When you boot your system, DOS sets the printer output to Lpt1. What the *mode* command does is enable you to set the parameters for Lpt1 or Lpt2. It allows you to specify the number of characters per line and the number of lines per inch for either Lpt1 or Lpt2. You have to use Printsw, however, to exchange Lpt2 for Lpt1. ▲

WordPerfect

"Congratulations on such an outstanding word processing software package! We, at Texaco, are really enjoying using your remarkable software system."

Richard W. Horchler, Computer Center Manager, Texaco

"Your system is light years ahead of any other word processor that we have tested."

Douglas L. Mayor, DL Mayor Corp.

"If you're a new PC owner who's scouting around for a top-of-the-line program, check this one out."

Lindsy Van Gelder, PC Magazine

"My favorite word processor."

Will Fastie, Creative Computing

SSI **SATELLITE SOFTWARE INTERNATIONAL**

TOLL FREE (800) 321-5906 TELEX 453-168 (801) 224-8554

AVAILABLE ON IBM PC, AND MANY OTHER MS-DOS COMPUTERS.

WE UNLEASH TH
POWERFUL GRAP

E WORLD'S MOST HICS TECHNOLOGY.

You'll never see Infocom's graphics on any computer screen. Because there's never been a computer built by man that could handle the images we produce. And, there never will be.

We draw our graphics from the limitless imagery of your imagination—a technology so powerful, it makes any picture that's ever come out of a screen look like graffiti by comparison. And nobody knows how to unleash your imagination like Infocom.

Through our prose, your imagination makes you part of our stories, in control of what you do and where you go—yet unable to predict or control the course of events. You're confronted with situations and logical puzzles the like of which you won't find elsewhere. And you're immersed in rich environments alive with personalities as real as any you'll meet in the flesh—yet all the more vivid because they're perceived directly by your mind's eye, not through your external senses. The method to this magic? We've found the way to plug our prose right into your psyche, and catapult you into a whole new dimension.

Take some tough critics' words about our words. SOFTALK, for example, called ZORK® III's prose "far more graphic than any depiction yet achieved by an adventure with graphics." And the NEW YORK TIMES saw fit to print that our DEADLINE™ is "an amazing feat of programming." Even a journal as video-oriented as ELECTRONIC GAMES found Infocom prose to be such an eye-opener, they named one of our games their Best Adventure of 1983.

Better still, bring an Infocom game home with you. Discover firsthand why thousands upon thousands of discriminating game players keep turning everything we write into instantaneous bestsellers.

Step up to Infocom. All words. No graffiti. The secret reaches of your mind are beckoning. A whole new dimension is in there waiting for you.

(For more information on Infocom games contact: Infocom, Inc., P.O. Box 855, Garden City, NY 11530.)

INFOCOM™

The next dimension.

For your: Apple II, Atari, Commodore 64, CP/M 8," DEC Rainbow, DEC RT-11, IBM, MS-DOS 2.0, NEC APC, NEC PC-8000, Osborne, TI Professional, TI 99/4A, TRS-80 Model I, TRS-80 Model III.

COMM LINES

by Kevin Goldstein

DSR, DTR, CD, and All That Jazz

The simple three-wire computer-to-modem interface described last month works great in theory but has a few problems in the real world. Briefly stated, that minimal interface leaves you flying blind as to the status of your telecommunications link.

Let's start at the beginning. It's time to transmit some data to a remote computer, so you fire up your PC, load your communication program, enter the number your modem's autodialer is to call, and sit back and wait. And wait. And wait.

That's when you realize you forgot to turn on the modem. That's not such an unlikely event—at one time or another, practically everybody has tried to use equipment that was off or unplugged.

To prevent such potential embarrassment, the RS-232 standard dedicates pin 6 of the interface connector to a signal called *Data Set Ready*, or *DSR*. (Modems used to be called data sets, but the term has fallen into disuse.) As you'd guess from its name, when DSR is *asserted* (true, or active), the modem is ready to send and receive information; if the computer monitors the status of pin 6, it will not send data to a modem that's turned off. As a matter of fact, if the cable between the modem and the computer gets disconnected, pin 6 will go false (inactive), again indicating the modem isn't ready.

Thus, pin 6 indicates that the modem is plugged in, turned on, and properly connected to the computer. (Well, probably connected properly. There's always room for strange goings-on in the land of telecommunications.) If you've ever wondered how your telecommunication program knows when to display that cryptic message informing you that your modem is turned off, now you know: It monitors line 6 of the interface connector.

Many interface signals have counterparts that travel in the opposite direction; DSR is no exception. If you were to go browsing through the remaining pins on the DB-25 connector, you'd stumble across a signal on pin 20 called *Data Terminal Ready (DTR)*. Just as you'd expect, Data Terminal Ready is a signal from the data terminal equipment (your PC) to the modem, telling the modem that the PC is ready to communicate.

Data Terminal Ready frequently has a different function, however. In the normal course of events, if you decide to originate a call for the purposes of telecommunicating, you load your communication program. Your program in turn sets (activates) the Data Terminal Ready line, telling the modem the computer is ready to begin. The phone is still hung up at this point; many modems respond to DTR's activation in this situation by switching to the originate mode. That interpretation is certainly reasonable—but not always appropriate; maybe the user is simply preparing for an expected call. Because of this ambiguity, most intelligent modems let the user specify originate or answer mode, overriding automatic selection based on the state of DTR.

The flip side of the coin is that when an autoanswer modem finds the phone ringing, it notifies the computer; if the computer then raises (activates, sets) its DTR, the modem answers the call—in answer mode, of course. Most modems don't answer the phone until DTR is set. That's a good arrangement; it keeps data from being lost while the called computer switches contexts and loads its communication program.

So far so good. This picture we're drawing (computer talks to modem, modem talks to phone line, phone line speaks to modem, and modem converses with computer) is starting to appear more failsafe. As it now stands, if the modem is turned off, unplugged, or improperly cabled, you'll know about it and communication will be halted; likewise, if a call is received before the computer is ready to accept data, communication will be put on hold. But what would happen if Spot chewed through the phone line while you were receiving data? You'd quit receiving data, of course, but rather than knowing you had a problem, you might just assume the other party had run out of data to send.

Does that problem sound vaguely familiar? If so, that's because it's exactly the problem we proposed to solve by having both near- and far-end modems transmit a constant carrier ("Comm Lines," September 1983). Indeed, if Spot did eat your phone cord, your modem would know immediately that the connection had been broken, since it would lose its receive carrier. Since the modem knows what's going on, it of course passes that information on to the computer, and that's why pin 8 of the RS-232 connector is reserved for *Carrier Detect*.

QuadDisk™ by Quadram™

Quadram™ presents QuadDisk.™ The new hard-disk system.

QuadDisk gives you maximum storage capacity in the limited space you've got. Turning your IBM™ Personal Computer, PC XT, Compaq,™ Chameleon,* or Eagle* computer into a micro powerhouse.

Choose from 6 to 72MB Fully Integral

QuadDisk comes in 6, 12, 20, 27, or 72 megabyte capacities. All integral to the computer for fast, convenient data access. And for added flexibility, there's QuadDisk with removable cartridges. For up to 6MB on-line storage plus added data security. (Not to mention virtually unlimited off-line storage.)

QuadDisk comes complete with disk controller board for total system integration.

Keep your personal computer personal

QuadDisk delivers all this power without turning your computer into a main frame. QuadDisk is compact. It installs in the same space as your floppy unit to become part of your system. Ready to work with your system.

By showing off its power, not its size, it keeps your personal computer personal.

Winchester technology made easy

But for all its power, this is one memory system that doesn't forget you're human. So QuadDisk comes with a virtual memory (RAM Drive) and a print spooler. Plus, QuadDisk's File Utility. To copy, rename, erase, and execute programs straight from a menu. Without using complex DOS commands.

Quadram Quality

Every QuadDisk comes stamped with the Quadram Quality seal. The mark of dependability and performance from the leader in microcomputer enhancements. So visit your local computer retailer today. Ask to see QuadDisk in action. When a little space is all you have, QuadDisk by Quadram is all you need.

QuadDisk features removable cartridges which give you unlimited off-line storage. It's Winchester technology made easy.

QUADRAM™ CORPORATION
An Intelligent Systems Company

4355 International Blvd. Norcross, Ga. 30093
(404)923-6666/TWX 810-766-4915(QUADRAM NCRS)

*IBM Personal Computer and PC XT are registered trademarks of the International Business Machine Corporation. Compaq is a trademark of the Compaq Computer Corporation. Chameleon is a product of the Seequa Computer Corporation. Eagle is a product of Eagle Computer, Inc.

As much storage capacity as you need, in as little space as you've got.

There's a little dance, called handshaking, that goes on when one modem calls another. Basically, what happens is that the modem that's been called (usually known as the answer modem) puts an answer tone on the line after going off-hook (after it "picks up the phone"). When the originating modem detects the answer tone, it puts its carrier frequency on the line. Both modems then set the Carrier Detect line to their respective computers. The presence of the Carrier Detect signal on pin 8 is extremely important. Take it away, and practically all communication circuits will refuse to accept any more data.

For most full-duplex lines, those five signals—Transmit Data, Receive Data, Data Set Ready, Data Terminal Ready, and Carrier Detect—are more than sufficient. For half-duplex lines, however, there's a signal missing.

Recall that on a half-duplex line, data can be sent in only one direction at a time. If the far end is transmitting data, the near end must wait before it can send its own data. That raises a problem: How does the near end know when the far end has finished transmitting?

The answer to that question unveils one of the major disadvantages of half-duplex lines, at least when they're used in a highly conversational mode. After the far end is done transmitting, the near end must wait a predetermined amount of time before it can transmit. That delay, known as *line turnaround time*, provides time for any echoes on the line to die down and for the near-end modem to determine that the far end's transmission has indeed ceased. (Phone lines echo almost as well as canyons, it sometimes seems.)

After the line turnaround time has passed, it is safe for the near-end computer to start transmitting. The modem notifies the computer of this fact by raising (asserting, setting to true) a signal known as *Clear To Send*, which occupies pin 5 of the connector.

While Clear To Send is an obvious necessity in half-duplex, it's something of a frill on full-duplex systems: If the computer operating in full-duplex is receiving Data Set Ready and Carrier Detect from its modem, it must be clear to send. It therefore might surprise you to learn that most full-duplex modems use the Clear To Send signal anyway. In some cases the CTS signal makes the job of monitoring the modem lines easier. If Clear To Send is asserted, the computer need not look at Carrier Detect or DSR. In other cases, without CTS, the computer might have to check all the lines individually. But in any case, these extra signals act as failsafes, and it defeats the purpose not to double-check everything available.

For low-speed (under 2400 baud) asynchronous full-duplex systems, these six lines (Transmit Data, Receive Data, Data Set Ready, Data Terminal Ready, Carrier Detect, and Clear To Send) carry *almost* all the signals needed to communicate competently. Several other pins on the DB-25 connector are used for synchronous systems; they have names like *Receive Clock* and *Transmit Clock*. (Synchronous systems live and die by clocks. See "Comm Lines," October 1983). Other lines carry test voltages.

There *is* one other signal used, but it doesn't really have much to do with the communication process; it just tells the computer when to answer the phone. Line 22 acts as a ring indicator; it's usually set up to pulse (turn on and off) when the phone rings. When the modem goes off-hook (is answered), line 22 stays active, so this pin could be considered a line-status signal (each modem has its own way of handling this line, but all conform more or less to the description just given).

With all that under our belts, let's return for a minute to the problem that was posed in last month's column: How do you connect your computer's asynchronous port directly to that of another, without intervening modems?

The answer suggested last month was gratifyingly simple: Just cross the transmit and receive lines. And, indeed, in a simple three-wire setup, that would work just fine. Just don't try it with your PC, because you won't get zip.

Why that simple-minded solution won't work with the PC is by now probably pretty clear: The PC uses the other signal lines we've discussed. While the earlier answer was right as far as it went, it failed to handle the Data Set Ready, Carrier Detect, and Clear To Send input lines to the PC. Let's look at some possible ways of handling those lines.

If your PC-to-PC direct link is to work, you've got to supply the first signal (DSR). Since there isn't any data set in your proposed link, the data set can't be ready; but the other computer can be. The cleanest way to handle DSR, then, is to connect the near computer's Data Terminal Ready to the far computer's Data Set Ready, and vice versa. (Near and far are misleading in this context, since both computers are local, but calling them computer A and computer B is too pedantic.)

Next in line (do ignore the pun) is Carrier Detect. Since there are no modems in this setup, you don't need to worry about Carrier Detect. Carrier Detect's purpose is to indicate that there's a modem at the other end and that the near-end modem can hear that other modem; a nearly equivalent function in the linkup we're describing now is knowing that there's a computer at the other end. If the signal being fed into Data Set Ready (which comes from the other computer's Data Terminal Ready) is active, then you know there's a computer at the other end and that it's ready to go. All you've got to do, then, is wire each computer port's Carrier Detect input over to Data Set Ready.

You can figure out on your own what to do with the Clear To Send input, but consider this: Your direct computer-to-computer hookup is a full-duplex operation, so if you've got Carrier Detect, you must be Clear To Send.

Right. String another wire between Clear To Send and Data Set Ready (by now a.k.a. Carrier Detect, a.k.a. the other computer's Data Terminal Ready).

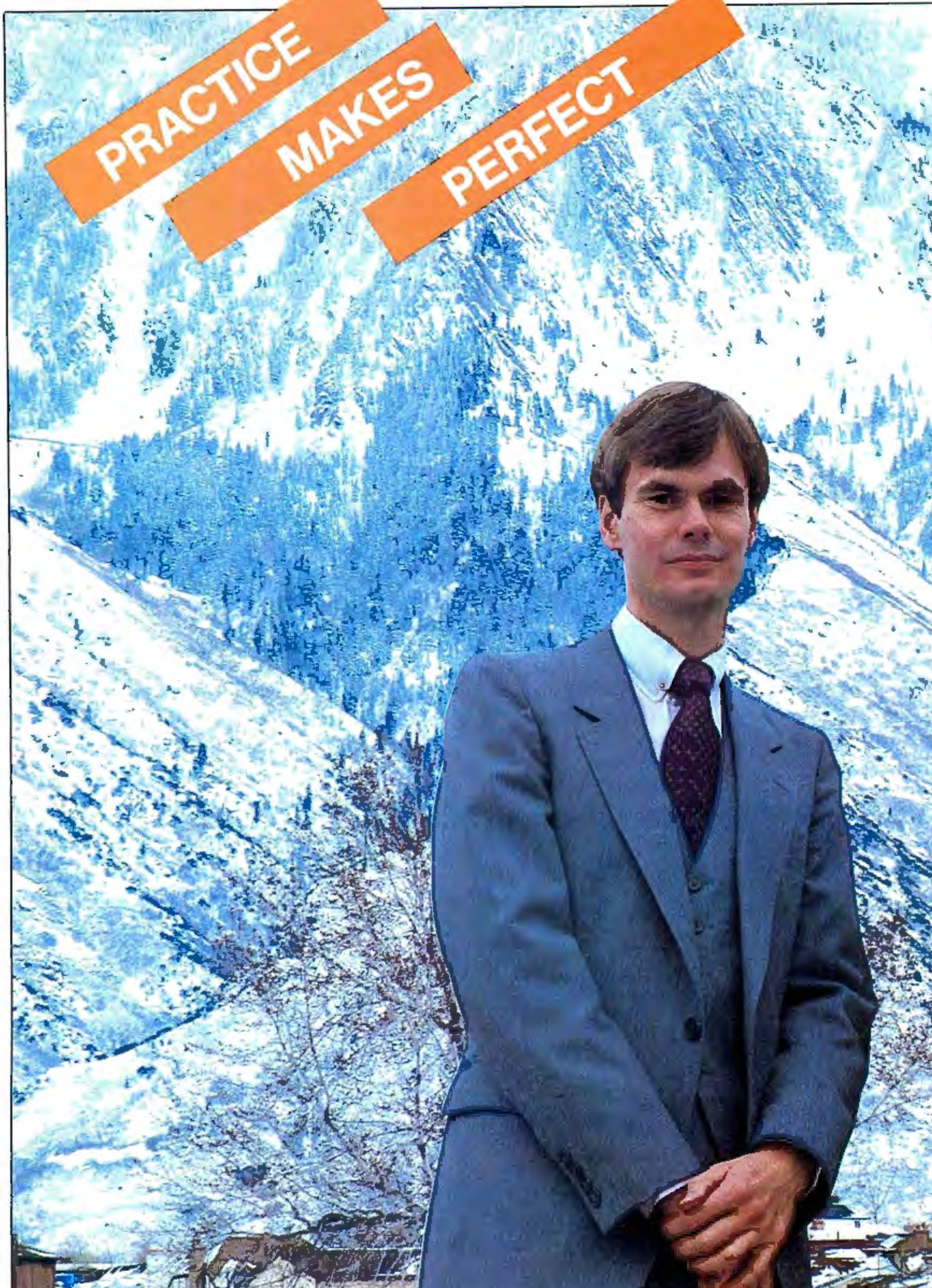
And there you've got it: a cable you could use to connect the asynchronous ports of two PCs directly. As a matter of fact, if you were to take apart a modem eliminator cable, you'd see the connections just described.

Not that stringing a wire between the input and Data Set Ready is the only way to do it. At either one of the two ports you could wire the Data Terminal Ready line right back into the Data Set Ready, Carrier Detect, and Clear To Send; then you would be back to your basic three-wire cable. That would clearly not be as robust a connection, however, because there'd be no way of knowing whether the far-end computer was even connected, let alone ready to receive data.

Incidentally, as you're glancing through spec sheets for modems, you're liable to run into some cryptic abbreviations for the various RS-232 signal lines. For example, the Transmit Data line is called BA and the Receive Data line is called BB. What the RS-232 standard attempted to do was divide all the signals into prioritized groups. The ground lines were thus called AA and AB; the next most important group was called BA and BB, and the modem control group names were all prefixed with C. Nobody remembered the abbreviations, of course, since they were almost totally unrelated to function, and completely unrelated to the more common descriptive names. Some modems use DTR as the label for an LED that indicates the status of Data Terminal Ready; others label the same LED DR. The abbreviations used in this column are the ones more commonly seen, but they are by no means universal.

Ah well. It's a pretty good standard otherwise. ▲

EXEC SSI: PRACTICE MAKES PERFECT



When a trumpet-playing college professor and the leader of the marching band got together back in 1975, they wrote a hit single. You can't dance to it but the words are great. It's *WordPerfect*, on the SSI label, and it's climbing straight to the top of the charts.

A Different Drummer. It all started eight years ago at Brigham Young University in Provo, Utah, where Bruce Bastian was a graduate student in music and director of the marching band. And quite a band it was. In those days, the Incomparable Cougar Band had more fans than the football team. Pete Peterson, vice president of SSI and a former student at BYU, recalls, "After the game, people poured onto the field by the hundreds and stuck around as long as the band was willing to play. Bruce did all the musical arrangements, and what he had was a 120-piece rock band, which to some people was totally out of character for BYU."

Bastian's future in music had never looked brighter—that is, until he was fired from the band director's job. You might say he was a victim of his own success. Under his direction, the Incomparable Cougar Band had gained national recognition. With all that attention, it just wouldn't do to have a *student* directing it. BYU went looking for a band director with credentials, someone with a doctorate in music.

Dejected about losing his position with the band, Bastian turned his attention from notes and measures to bits and bytes. No stranger to computers (two of his brothers worked for IBM), he'd written a graphics program that helped him plan the band's marching formations. He could march the band around the screen and get views from every angle—the 50-yard line, the end zones, the Goodyear Blimp, even a gopher's view from underneath the field. Once he worked out the formations, the program took it from there and printed out marching instructions for everyone in the band.

With program in hand, Bastian went to see Alan Ashton, someone who he knew could appreciate an elegant program as much as a Mozart concerto.

Ashton had been teaching computer science at BYU since 1972, the year he received his doctorate from the University of Utah in Salt Lake City. While there, he'd studied under the likes of Dave Evans, founder

BY TERRY TINSLEY DATZ AND F. LLOYD DATZ

SSI cofounders Bruce Bastian and Alan Ashton: "We're always asking ourselves how computers can be made aesthetically pleasing."

of Evans and Sutherland (you've seen their graphics in *Star Trek II*) and Bob Barton, designer of the Burroughs 5000. A musician at heart, Ashton had worked his way through school giving trumpet lessons and making the rounds with local dance bands.

He thought Bastian's program was elegant, all right. He was so impressed with it that he persuaded the musician to switch camps. Cutting his ties with the music department, Bastian went on to get a master's degree—but in computer science, not music.

A drastic change, you say? Not to Bastian. "I've always thought that writing a program is a lot like writing a piece of music," he says. As a fellow musician, Ashton agrees. "We're always asking ourselves how computers can be made aesthetically pleasing."

Unfinished Symphony. When he completed the master's degree in the fall of 1978, Bastian's list of job offers would have made any graduate's head spin. He could have chosen to work for, among others, IBM, Hewlett-Packard, Digital Equipment, or Texas Instruments. But he had other plans. For one thing, he had family ties in Utah. For another, he was set on working with Ashton, with whom he'd become close friends. His mind made up, he turned down all offers and joined with Ashton and a financial backer to start a software company called Satellite Systems, Inc.

Like many other new software companies, this one was short-lived. The financial backing promised by the third partner never came through, and Satellite Systems fell from orbit within a month.

Ashton still had his position at BYU, but Bastian found himself holding a mortgage on a brand-new house with no job in sight. This was an ironic turn of events, considering that just a month ago he'd turned down more offers than Reggie Jackson.

With Ashton's help, he scoured the Provo area and landed a programming job with Eyring Research—barely in time to make his first mortgage payment. In the meantime, Eyring had contracted with Orem, a small town near Provo, to write a word processing program for that city's Data General minicomputer. As it happened, no one at Eyring knew the first thing about word processors.

What Bastian didn't know about word processing he was willing to

learn. And what better person to learn from than Ashton, who had authored several program editors and who had recently designed a word processor on paper. With the hope of resurrecting SSI in the future, Ashton agreed to help with the project. They signed an agreement that the two of them, not Eyring, would own the program's design when it was finished.

For the next six months Bastian worked on the word processor as an Eyring employee. Ashton kept his position at BYU but contributed his free time, evenings and weekends.

Play It Again, Sam. In May of 1979 the people at Eyring decided that the new word processor met the specs for Orem City. When they asked Bastian to move on to a new project, he moved on all right—he quit Eyring and, for the second time, became business partners with Ashton. Again they named their company SSI, but this time the letters stood for Satellite Software International, an impressive name for a company with just two employees. Bastian became president, and Ashton, who kept his position at BYU, executive vice president. Their plan was to market their word processor, which they called *SSI *WP*, to other Data General customers.

Why the name Satellite Software International? Smiling, Bastian admits, "One of us had seen the letters SSI on an abandoned boxcar and liked the way they looked. We just put words to the letters and came up with Satellite Software International." Such a name has its drawbacks. "Occasionally we get calls from people wanting to buy television satellite dishes. We're not crazy about the name. I guess we'd quietly change it if we could."

The second time around Bastian and Ashton managed to do the impossible: They got SSI off the ground without the benefit of investment capital. To raise the funds to cover their initial expenses, they sold *P-Edit*, a program editor based on their word processor, to Itel.

Such financial conservatism is characteristic of SSI. "We've never expanded without using money from another project," Ashton explains. "Offices and equipment have always had to wait until we had the money in hand."

Over the next few months, they polished *SSI *WP* . With no money in hand for an office, much less for a computer, they borrowed Orem's city offices and worked at night and during odd hours. In return, it was agreed that Orem would receive any program enhancements for free. During this time, the Ashton and Bastian families were just about the only believers in SSI's future. Bastian recalls that even a few family members were die-hard skeptics who advised them to be realistic and get a regular nine-to-five job. The people at Eyring were so sure of SSI's impending failure that they were still holding Bastian's job open for him—and there were many times he was tempted to go back.

Ashton, too, remembers that these weren't easy times. "We worked long hours. We'd write code until two or three in the morning, or as long as one of us could stay awake." They kept up this exhausting schedule for six months, working every day of the week except Sunday and taking no holidays except Christmas Day.

SSI's vice president and chief orchestrator Pete Peterson sees himself as the company's humanoid, who keeps "both feet on the ground."

By January of 1980, the word processor was ready to go. SSI finally opened an office and began to market the program to General Instruments and other Data General users. With a nonexistent advertising budget, Bastian and Ashton got the word out, the hard way, by manning the phones and licking the stamps themselves.

By the end of that year, the word had indeed spread. Their program was rapidly becoming *the* word processor for Data General Equipment, and SSI was doing well enough to hire some help at last. They hired Pete Peterson to manage the office and took on two additional programmers.

Variation on a Theme. Had the PC not been introduced, SSI's success story might have ended here—no modest success, to be sure. But no sooner had IBM made its announcement in August 1981 than the tempo at SSI went from *andante* to *allegro*.

After all, if SSI had a word processor that performed every bit as well as a standalone, why not adapt it to run on the PC? Encouraged by some Data General users who wanted to switch to the PC, Bastian and Ashton set off for Salt Lake City and ComputerLand, where they latched on to one of the first PCs in Utah.

Having set up their PC on some boxes in the back of the office, they went to work on the program conversion. In those days there was no assembler available for the PC, and neither Microsoft nor IBM was inclined to talk to a startup software company in Orem, Utah. But Ashton says he and Bastian weren't complaining. "We've always been comfortable writing in machine code anyway. We like to stay close to the machine."

The conversion went smoothly enough, and by November 1982 their old friend *SSI*WP* had been converted, repackaged, and renamed *WordPerfect*. Within six months, the PC version of *WordPerfect* had become SSI's number one priority. A year later, on its first birthday (after a steady climb up the charts), it was the first word processor to nudge ahead of *WordStar* on *Softalk*'s Top Thirty. What better birthday present could a software company ask for?

With *WordPerfect* in the spotlight at SSI, Data General's word processor now plays second fiddle to the IBM version; ironically, the minicomputer cousin gets its updates as hand-me-downs from the PC.

Critic's Choice. Bastian and Ashton attribute their success to many factors—their unwillingness to compromise on quality, a spirit of cooperation, hard work, family support, and a little luck. Above all, they feel that *WordPerfect* speaks for itself in terms of quality. "We aimed high from the beginning," Bastian explains, "and we kept our standards high all along."

Of the word processors adapted for the PC from other machines, *WordPerfect* was unique in that it hadn't ever run on a microcomputer before. Bastian feels that this gave *WordPerfect* an edge. "Other word processors were scaled up to run on the PC. We scaled ours down."

From the beginning, he says, their main concern was to make the program powerful without letting that power get in the way. How did they do it? By seeking the input of secretaries at the Orem City office. What better critics than people who do word processing day in and day out?

Bastian concentrated on the human interface, writing the code at night and testing it on the secretaries the next day. "I originally put a ruler at the top of the editing screen but dropped it because the secretaries thought it cluttered the screen. That's how we ended up with such a clean screen." The spelling checker was also adapted to the way secretaries work. Bastian says they liked having the option of checking an individual word, a page, or an entire file—all without having to exit from the word processor.

"Above all," he says, "we measured the frustration level. *WordPerfect* is an extension of your fingertips. You can use it all day long without going berserk. That's something that can't be said for a lot of word processors."

Ashton agrees that the human interface has to come first. He remembers an experience from his college days. "A group of us sat around the computer lab one evening, listening to a Bach fugue while the computer entertained us with some graphic displays. On our way out of the building, we passed the university's main computer room. Here were people scurrying around serving the computer—what a contrast to what we'd just experienced upstairs. I've always thought that

people should be the masters of computers, not the other way around."

Their high standards paid off in the way the Orem employees responded to the word processor: They loved it. "Anytime anyone in government likes something," Bastian says, pounding the table, "it *has* to be pretty wonderful."

Not all of *WordPerfect*'s features can be traced back to input from the secretaries at Orem. Some, Bastian says, were the result of struggling to bring a new product to market with no support staff. When the time came to send out advertising mailers to Data General customers, for example, SSI tacked on a mail merge feature to get the job done. Then, instead of buying accounting software, they added math capability; even today, SSI stores all its monthly accounts inside *WordPerfect*.

Being self-sufficient has other advantages. "We use *WordPerfect* every day, so we won't put up with a feature that doesn't work the way we want it to," says Bastian. "If something's not easy, elegant, and powerful, it just won't be done."

Ashton is convinced that another reason for *WordPerfect*'s success is that SSI has continued to be closely held. "If we'd gone to capital investors, all of them would have wanted something different. Profit, not quality, would've been the bottom line and, in the end, *WordPerfect* wouldn't have turned out as it did. We like the freedom we get from having control."

Perfect Consonance. But do the two of them ever disagree about where SSI should be headed? Rarely, it seems. In this case at least, opposites seem to have attracted. Soft-spoken and reserved, Ashton provides the perfect counterpoint to his outspoken partner's enthusiasm. "Alan and I are in good harmony," says Bastian. "It's the employees that sometimes disagree with us." Then, grinning, he adds, "And often they end up winning."

One of SSI's forty-three employees who end up winning much of the time is Pete Peterson, vice president and chief orchestrator. He sees himself as SSI's humanoid. "Someone has to keep both feet on the ground and make sure the bottom line is always on the plus side," he says. Other winners include SSI's three marketeers: Dan Lunt, director of marketing; Andre Peterson (no relation of Pete's), who works with dealers; and Doug Lloyd, liaison for SSI's three-piece-suit accounts.

It's not just musical talent that runs in the Bastian family. Bastian's two brothers, Lewis and Reese, have given up their positions at IBM and now do their programming at SSI. Ashton says that finding other programmers hasn't been much of a problem. "We get plenty of resumes, but we go out looking for the people we want." They haven't had to look very far. His computer science classes at BYU have been a continuing source of bright new talent.

Having all that talent around makes it easy for SSI to do everything in-house. "We take the product from conception to shelf," Ashton says proudly. "We've found that we can do things better ourselves." The SSI do-it-yourselfers write and typeset the manuals, duplicate disks, design the artwork, compose paste-ups, and shrink-wrap the final product. Oh yes, and they also provide product support, something that SSI takes very seriously.

Other Performers. *WordPerfect* may be the star of SSI's show, but it's not the only performer. For starters, there's *WordPerfect Sorter*, a program that sorts name and address records created with *WordPerfect*. *WordPerfect Speller*, formerly sold separately, is now bundled with the word processor. There's also *P-Edit*, a program editor that shares many of *WordPerfect*'s features, and *SSI Forth*, an interface, editor, and debugger for programming in Forth. *SSI Legal*, a program that handles scheduling and billing for small legal offices, is currently in the process of being overhauled. SSI also markets a utility program that converts *WordStar* files to *WordPerfect* format, and it'll soon be providing the same service for Wang files.

The latest addition to SSI's software family is *Personal WordPerfect*, a less expensive, slightly scaled-down version of *WordPerfect*. It's no coincidence that this baby's in-house name has always been "Junior." Ashton admits to having the PC*jr* in mind all along. Needless to say, *Personal WordPerfect* runs as is under DOS 2.1, PC*jr*'s operating system, as well as under DOS 1.1 and 2.0. It's even in the process of undergoing a name change—to *WordPerfect Junior*, what else? As Junior ages, he'll become more powerful, taking on mail merge capability and an optional spelling checker.

Future Performances. SSI is about to raise the curtain on a spreadsheet (*MathPlan*) and a database (*DataPath*). SSI doesn't plan to promote sales through heavy advertising. As they did with *WordPerfect*, personnel will visit stores, give demonstrations, and let the products sell themselves.

What about the mouse? "It's not that important for word processing," says Bastian. "Its use depends entirely on the package and what you're trying to do. With graphic applications, the mouse is essential. With word processing it's less helpful since you're not drawing pictures."

According to Ashton, the day is coming when word processors *will* be drawing pictures. "Word processors will have graphics editors. They'll be able to do complex pictures, not just pie charts and bar graphs. If you want to draw a house with trees, for example, you'll be able to." When that day comes, he says, "*WordPerfect* will be there drawing with the best of them."

Will *WordPerfect* continue to get top billing at SSI? "Everyone needs to type," says Bastian, "so *WordPerfect* will remain the hub of our software family. We never wanted to be a vertical software house. We'll stay in horizontal marketing."

One thing is certain: *WordPerfect* is an ongoing project. "I don't think *WordPerfect* will ever be finished," he says. "If it is, I guess SSI will be finished too." ▲

Next April 15th, you could be adding up your golf score instead of your taxes.

With your IBM PC and Best Programs' TaxCut™ program, you can use the long days and nights you used to put into preparing your taxes for something more enjoyable.

TaxCut includes two second-generation programs, one for tax preparation and the other for tax planning, pioneered and refined by tax and programming experts—and by PC users like you. With step-by-step, on-screen prompting, the tax-preparation program helps you prepare and print the 1040 long form and more than 30 other commonly used forms and schedules.

You don't have to know all the regulations. You don't have to figure out IRS instructions. If you need help, the on-screen prompter tells you exactly which page in the comprehensive reference manual has the answer. You won't waste time answering questions more than once because the program automatically transfers information from one tax form to another.

TaxCut also includes a tax planner program that allows you to decide for yourself whether or not to set up an IRA, what effect a new mortgage will have on your tax liability, and the tax implications of a wide variety of other financial alternatives.

TaxCut is compatible with the IBM PC, the PC/XT and the COMPAQ computer. The program requires at least 128KB memory and one double-sided diskette drive. Add Best's Professional Finance Program (PC/PFPII), which tracks and computes data for input into TaxCut, and you have a complete financial and tax package.

You can have the whole, proven, second-generation tax-preparation and planning package for $255.00. This price includes extensive customer support and a newsletter to keep you up-to-date. And both the cost of the program and the actual cost of doing your taxes with it are tax-deductible.

The TaxCut program is available for immediate delivery. Call us toll-free at 1-800-368-2405 for more information.

Next April 15th, you'll be very glad you did.

BEST PROGRAMS

"The Quality Software Company"

SYSTEM Notebook

by Alan Boyd

Graphics, Assign, Recover

Welcome to another year of "System Notebook." Over the past few months we've looked at the new commands that came with DOS Version 2.0. Last month we looked at the last of the internal commands new to DOS; this month we'll continue by looking at some of the new external commands.

Graphics. The *graphics* command is one of the nicest of the new DOS commands. If you have a PC equipped with a color/graphics adapter and an IBM-supplied dot-matrix printer, or an Epson-supplied equivalent, you will be able to produce several different types of graphics dump with the *graphics* command. (A graphics dump is simply a translation of the bits that make up the screen into a string of bytes that will produce a similar display on the printer, reproducing the screen dot for dot on the paper.)

The IBM manual is woefully lacking in details about *graphics* (as it is about most of DOS's other fun commands); it covers the entire command in only one page. What you may not be able to gather from reading the manual is that *graphics* does nothing more than install a driver in DOS allowing the PC to work with the IBM or Epson dot-matrix printer equipped with a graphics option. Without the additional ROMs the graphics option gives your printer, you will be able to use only the text-printing features of the command. Without the command, you'll be unable to dump graphics through DOS, no matter what your printer's capabilities are.

Assuming that you do actually have the proper printer option, enter

```
A>GRAPHICS
```

on the keyboard. That's all there is to it.

If you haven't issued the *graphics* command, the only thing you can dump to your printer is the text screen. You can do that at any time anyway, by pressing shift and print-screen simultaneously, but if you have a graphics image on your screen and you haven't issued *graphics*, you won't get much of a printout.

You can get an idea of how shift-print-screen dump works by booting with the DOS master disk and, when the A> prompt appears, entering the following command to load Advanced Basic:

```
A>BASICA
```

You will see Basic's "OK" prompt and a message indicating how much memory you have available (it should be something on the order of sixty thousand bytes). Make a note now of how much free memory Basic says you have.

Replace the DOS master disk with the "Supplemental Programs" disk containing all of the Basic sample programs, many of which produce graphics. Enter

```
RUN "ART"
```

At any point while Art is running, try pressing the shift-print-screen combination; observe that only the characters displayed by the program are printed: "press ESC key to exit" and "The City." The program's graphic display is not printed. Now press the escape key to return to Basic and type *system* to return to DOS.

Once you're back in DOS, replace the DOS master disk in drive A and invoke the *graphics* command. Nothing apparent will happen. However, now that you have installed the new graphics driver, you are ready to retry the Basic Art program.

If you have a system with less than 96K of memory, you'll note that when you return to Basic, the amount of free memory will be less than it was before because of the memory consumed by the printer driver. Step through the Art program the same way you did before. This time, however, when you press shift-print-screen, you should be rewarded with a printout of the graphics screen, captured at the instant you pressed the keys. While the graphics screen is printing, the Basic program remains frozen; it returns to action as soon as the dump is completed.

With the graphics driver installed in DOS, you can print two different types of graphics dumps corresponding to the two graphics modes that are available and supported by Basic. The first mode—320 by 200—is the one that you have just seen in the Art program. This mode makes four colors available on the video display. When you dump a screen in this resolution to the printer, the driver assigns a different shade of gray to each color.

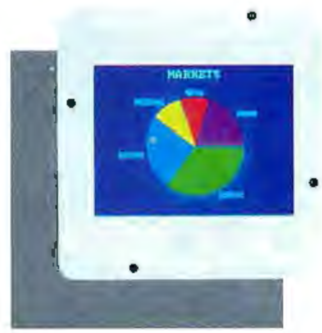
The *graphics* command also supports the 640-by-200 high-resolution mode, which has only two colors available on the video display—usually black and white. When you tell the graphics driver to dump a high-resolution screen to the printer, it turns the image sideways so that the full resolution of the printer can be put to best use. This also keeps the aspect ratio (the ratio of height to width) correct, so that circles appear as circles rather than as ellipses.

As you'll notice, the graphics driver makes no attempt to use the maximum resolution of the printer. It simply takes the dots on the screen and multiplies them to make bigger dots for the printer. Each of these bigger dots is actually a solid rectangle consisting of several smaller dots. This is not exactly the best method of printing out graphics with a printer that has resolution as high as the IBM or Epson, but it does come free with DOS.

Like many of the other PC-DOS commands that are not part of the generic MS-DOS operating system, the *graphics* command seems to have come from somewhere other than Microsoft. Essentially a driver, it does not seem to follow the rules that DOS lays down for installing device drivers. It also does not work very well with Basic, which, since it has good graphics capabilities, would benefit tremendously from the *graphics* command's capabilities. To make the command work with Basic, you must either catch the graphics display on the fly (as you did by hitting shift-print-screen) or insert a special command in the program to halt the graphics display. With the program stopped, you can dump

It's easy to see why DEC, Hewlett Packard, NEC and Burroughs chose

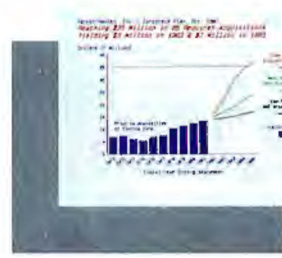
Graphwriter® for presentation graphics.

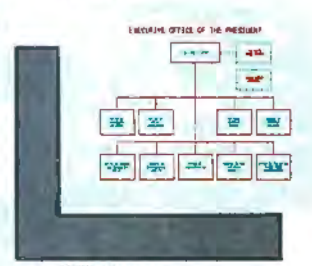


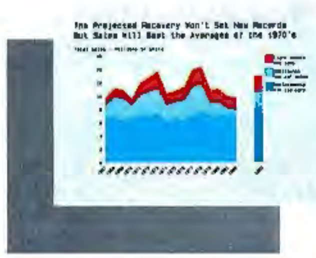
When Digital Equipment Corporation, Hewlett Packard, NEC Information Systems, and Burroughs Corporation selected a presentation graphics package to offer with their microcomputers, they surveyed the market carefully. All of them made the same decision: They chose ***Graphwriter.***

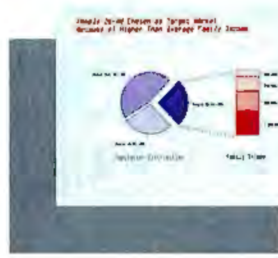
When you choose a presentation graphics package for your personal computer, you will look for the same capabilities these major manufacturers looked for. Highest quality output. More chart types. Flexibility to create just the look you want. And easy enough for everyone in your organization to use. Only ***Graphwriter*** can give you the power of a mainframe graphics package on your microcomputer.

It's easy to see why ***Graphwriter*** is the standard for presentation graphics—for hardware manufacturers as well as personal computer users like you.









Graphwriter versions for the DEC Rainbow™, HP150™, NEC APC™ and Burroughs ET2000™ are available from the manufacturer or an authorized dealer. ***Graphwriter*** for the IBM PC/XT is available through IBM dealers, including ComputerLand and Entre Computer Centers.

IBM PC and XT, Rainbow, HP150, APC and ET2000 are trademarks of International Business Machines, Corp., Digital Equipment, Corp., Hewlett Packard, Corp., NEC Information Systems and Burroughs, Corp., respectively.

For more information, contact:
Graphic Communications, *Inc.*
200 Fifth Avenue
Waltham, Massachusetts 02254

617 890-8778

More graphic formats
column (vertical) charts
stacked column
clustered column
bar (horizontal) charts
stacked bar
clustered bar
single pie
double pie
triple pie
quadruple pie
proportional pie
line charts
scatter plots
bar-line combination
text/word charts
Gantt
organization
bubble
table
pie-bar combination
surface line
stacked line
line-table
double stacked bars
range
paired bars
overlay text

More flexibility
move chart elements
heading justification
footnote justification
italic fonts
bold fonts
bold lines
variable font size, all text
variable font color, all text
values displayed on bars
values displayed for pie slices
adjustable bar widths
adjustable pie sizes
pie rotation
pie placement
single bar color control
single bar omission
unequal line lengths
8 line types
4 line marker types
8 line types
5 frame options
8 fill patterns
axes labels ($, % or x)
log scaling
semilog scaling
multiple curve fits
floating comments

More space for your message
150 character heading
150 character footnote
150 character comment
50 character axis titles
40 character legends
20 character bar/slice labels

More ways to present
paper
transparencies
35mm slides
coated paper plotting
horizontal page plotting
vertical page plotting
multiple plots per page
custom size and location
graphics only plotting

Easier to use
individual format programs
pre-designed formats
format selection from chart book
input forms
spreadsheet file access
data transfer between formats
format-specific help messages
prompt-specific help messages
user-defined default options
screen preview
preview plotting
printed proof sheet
"built-in artist"
automatic chart composition
automatic page layout
pen sorting
minimal pen changes
pen change prompting
faster solid fill
transparency color separation
automatic storage of last data
chart storage
template storage

the screen and then restart the program.

If you need a graphics dump program, then the *graphics* command will probably fill your needs. However, if you are looking for something more sophisticated, there are several packages on the open market that may better meet your needs.

Assign. The *assign* command seems to have been designed to overcome a frustrating limitation of programs that have all the disk drive information written into the software and the hard code, forcing you to use drive A for program information and drive B for data. This may have worked just fine on the programmer's system, but may not be appropriate for you. You may have, for example, an external hard disk drive set up as drive C. Perhaps you want the hard disk to receive data but you can't do this because the programmer has set up and hard-coded the drive designations.

The simple little *assign* command provides you an out by letting you indicate that any drive in your system should behave as though it were physically connected to any other slot. Thus you can assign all information the application program sends to drive B to the hard disk in drive C:

```
A>ASSIGN B=C
```

Note that there is no need to include the colons usually associated with the drive designation letters. In fact, if you attempt to include the colons you will be rewarded with an invalid parameter error message. One clue to this strange behavior is the fact that the *assign* command is not generally distributed with versions of MS-DOS for other computers. This command (along with a few others that seemingly break the DOS rules) apparently didn't come from the same source as the rest of the operating system; it may have been added as an afterthought.

However, if you use the correct form just shown and then look at drive B's directory, you'll see a listing of the files on drive C. The directory listing will inform you that the information came from drive B by the usual

```
Volume in drive B has no label
Directory of b:\
```

directory listing header, proving that even DOS believes that the information is actually coming from drive B instead of drive C. In fact, if you do a directory of drive C, you will be given exactly the same listing but with a B replacing the C in the header.

"Aha," you might say. "This is a way to fool DOS. If I copy a file from drive B to drive C I could get two files on the same directory with the same name."

Not so. If you do tell DOS to copy a file it will simply overwrite the original. The result is that you will still be left with the same file.

One of the perils of the *assign* command is what happens when you boot your system with the DOS master disk and then decide to try out the command for yourself. Of course, since most people have systems with two disk drives, the most logical choice is to assign A to B by issuing

```
A>ASSIGN A=B
```

Doing this places you in an unrecoverable situation, since the *assign* command is an external command. The only way out of this is to switch disks physically, placing the master—or any other disk with a copy of the file Assign.com—into drive B. Alternatively, you can reboot.

Use *assign* sparingly. It apparently has some strange side effects when issued in conjunction with the *print* command. That's a combination to be avoided, as you'll see next month.

In fact, the *assign* command should generally be avoided. It's shipped with DOS for the benefit of those inconvenienced by sloppy programming. Programmers developing software to run under DOS should be aware of the following quote from the IBM DOS manual:

> "If you will be developing an application program, we recommend that you avoid using specific drive assignments within your program, but instead allow the user to specify the drive(s) to be used."

Wise words indeed. If all programmers would follow this simple rule, there would be no need for commands such as *assign.*

Recover. The *recover* command is one of those DOS commands you hope you never have to use. Its purpose is to reconstruct a damaged file from the information contained in the directory and the file allocation table. It is a brutal command that has saved many a day and proved ineffective on many others.

DOS keeps track of files by using a directory linked to the file allocation table. From this information, DOS knows exactly where on the disk each file is located. During the file-loading process, DOS looks up these tables and retrieves each of the disk sectors that contain a part of the file. It then strings them all together in memory and re-creates an exact duplicate of the file as it was originally created. The problems start when DOS finds that one of the sectors has become damaged.

Any time DOS encounters a sector it cannot read, it gives you the option of retrying, ignoring, or aborting the attempt. If you retry over and over again and DOS fails to read the sector, then you have a bad sector on your hands.

Bad sectors come in two types. The first and more devastating is a destroyed sector that's actually part of your file. If DOS can't read it, then there's little that any program can do to salvage such a sector. The other kind is a sector that's incorrectly represented in the directory or file allocation table (FAT); DOS can, on most occasions, correct this kind of problem, by reconstructing the directory or FAT.

In the first case, where a sector containing part of the file is damaged, the *recover* command can only do so much. After all, if the

sector can't be read, it can't be read. Those data are lost, and *recover* can only attempt to salvage the rest of the data in the file. If the file in question contains program data, the entire program is likely to be unusable. And if the file contains any kind of structured information—formatted records from a database, for example, or encoded or compacted data—then the recovered file will probably be useless.

Recover's greatest use is in saving text files. The data in the bad sectors will not be recovered, but the rest of the file should be okay. Therefore *recover* does little more in this instance than you would do by opting to ignore any file-reading errors when you discover them.

If you suspect that a file has become damaged, you can try to use the *recover* command as follows:

A>RECOVER Badfile

where Badfile is the name of the corrupted file. *Recover* will do its best to reconstruct the file from the pieces it can read. The bad sectors it finds will be marked as used, so they won't be reallocated to another file later. The net result is that a second file with the same name as Badfile will be created. A recovered text file will be fully readable, but it will not contain any of the information that had been stored in the bad sectors.

The other, more useful application for *recover* is the reconstruction of directories. Imagine, if you will, the following scenario. You have constructed a hierarchical directory structure as follows:

```
            ROOT
USR1        USR2        USR3
FILES       FILES       FILES
```

and, in the directory \USR3\FILES you have the following files

LETTER.MOM
INCOME.82
INVENTRY.LST
INSURNCE.LST

These files all contain valuable information.

However, one day you discover you cannot log on to the directory \USR3. You should remember that DOS stores any directories you create as normal disk files; these directories are subject to wear and tear, just as any other disk file is. A directory may, in fact, be more damage-prone than an ordinary data file, because you access directories so often. Obviously, if you cannot log on to the \USR3 directory, then all of the information stored in that directory and all directories beneath it is inaccessible.

It's in a situation like this that *recover* earns its living. If you discover a problem with a directory, then the *recover* command entered without a filename argument will reconstruct the files on the disk.

One of the basic assumptions *recover* makes is that none of the information in the directory is correct. So, it throws it all away, including any subdirectory structures that you have built, and starts from scratch—meaning that it throws away all the names that were initially assigned to files and subdirectories alike. In fact, since *recover* has discarded all the original directory information, it has no way to make a distinction between actual files and subdirectories and will treat them both as regular files. It then proceeds to reconstruct a new root directory containing all the files that were on the disk, using the scratches of information (such as the FAT) that it has available to work with.

As *recover* reconstructs the file directory, it arbitrarily assigns new names to the files it recovers. These files are given names following the

FILEnnnn.REC

scheme, starting at File0001.rec and increasing. Once the directory has been reconstructed, you will need to rename the files to their original names. Although this is inconvenient, it does result in the recovery of valuable data.

You may want to see the *recover* command in action in a simulated situation, just in case you ever have to rely on it. You can do this by constructing a simple three-level tree and placing a single file at the bottom. First format a blank disk and put it in the B drive. Then log on to it and enter the following sequence of commands:

```
B>MKDIR DIR1
B>CHDIR DIR1
B>MKDIR DIR2
B>CHDIR DIR2
B>MKDIR DIR3
B>CHDIR DIR3
B>COPY CON: TESTFILE
This is a file.
^Z
```

This creates the three-level directory tree and places a solitary file named Testfile at the lowest level. If you remain in the Dir3 directory and list its contents, you will find that Testfile is about seventeen bytes long.

After you have done this, issue the *recover* command. Since this is an external command, you'll need to have the DOS master disk or a duplicate in drive A. Enter the following command:

B>A:RECOVER B:

to which DOS will respond:

Press any key to begin recovery of the file(s) on drive B

As soon as you press a key, the recover program will go to work. It takes a surprisingly short time, and when it is done it will report how many files have been recovered. In this case you'll see:

4 file(s) recovered

This report may surprise you, since you only created one file. But, of course, the directories Dir1, Dir2, and Dir3 are considered by DOS to be files and, in fact, are converted by *recover* into ordinary files. If you take a look at the new root directory that *recover* has created, you will see something like the following:

```
B>DIR
Volume in drive B has no label
Directory of B:\
FILE0001 REC      512    1-01-80   4:53p
FILE0002 REC      512    1-01-80   4:53p
FILE0003 REC      512    1-01-80   4:53p
FILE0004 REC      512    1-01-80   4:53p
        4 File(s)   177664 bytes free
```

You can see that *recover* has taken each of the files on the disk and created a new root directory with all the files in it, including those that were previously directories. It cannot assume that the information in the supposedly damaged directory is correct, so it redoes everything.

The problem now is to determine which file is which. The *type* command offers the best means of accomplishing this. Simply type the contents of each file onto the screen. It will be immediately obvious that the recovered files 1 through 3 are what remains of the old subdirectories. File0004.rec is the file you had created earlier.

This raises the issue of file size. When you initially created Testfile, it was only about seventeen bytes long. The recovered version is now 512 bytes. Where did all the extra space come from?

The *recover* command will, in most cases, return a fixed-up file that is longer than the original. This is because it works with blocks of disk space, each of which is the equivalent of the DOS allocation unit. You may remember that when storage space is required, DOS parcels it out in allocation units or blocks; the allocation unit is the smallest unit that the Recover program can work with.

This extra space at the end of the file will often contain random, meaningless data. In the case of text files, it will probably be necessary to edit the file and remove everything that has been added. This shouldn't be onerous, though, considering the alternatives.

As you can see, the changes the *recover* command makes to files and directory structures are drastic. It is best to salvage whatever information is still intact before using *recover*. Here are some basic steps you should take when you discover a disk error:

1. Try to load the damaged file several times before concluding that you have a problem. Try several different methods of accessing the file (*copy, diskcopy, type*, and so on).

2. If step 1 fails, try putting the apparently damaged disk on another drive and accessing the file there.

3. If steps 1 and 2 fail, then you probably have a damaged sector. See if you can access other files on the same disk.

4. If step 3 fails, then you have a bad directory sector. This spells trouble. Proceed to step 6.

5. If step 3 is successful, immediately copy all of the files you can access onto a backup disk. Then delete the originals. This will make for fewer files on the recovered disk and will help you identify which files are which. Use the *recover* in the individual file mode (*recover Badfile*).

6. Your disk is seriously damaged and you must use the full power of the *recover* command (*recover B:*). After you've done that, you'll have to search through the rubble and see what you can preserve. You may never need to use *recover*. But digital accidents do occur. The existence of the command should in no way be considered an endorsement of sloppy media management. The best way to recover from an accident is to pull out your backup disk and use it instead. All important data should be backed up and verified frequently. There simply is no alternative to good backup habits. ▲


WE'VE CREATED
A WHOLE NEW BREED
OF INTEGRATED
SOFTWARE
FOR THE IBM PC/XT.

NOW, ALL IT TAKES IS ONE SCREEN TO DO EVERYTHING YOU'VE ALWAYS WANTED TO DO, AT ONE TIME.

Word processing. Spreadsheets. Data base management. Charting. JACK2 is the first integrated software product to do them all, simultaneously, on the same screen. All without ever changing disks or exiting programs.

No need for windows. No need for additional monitors or hardware of any kind. No ueed to close one file before you open another. And no need to learn a specialized computer language.

JACK2 is as easy to master as it is powerful to use. Picture a screen that graphically displays your disks and names them. With envelope icons that can be scrolled up or down from 1 to 50 showing you all your files. JACK2 will even show you the forms inside your envelopes, then let you choose the one you're looking for simply by pointing to it. All commands are in English. All are displayed on a single line and all have the identical function throughout JACK2.

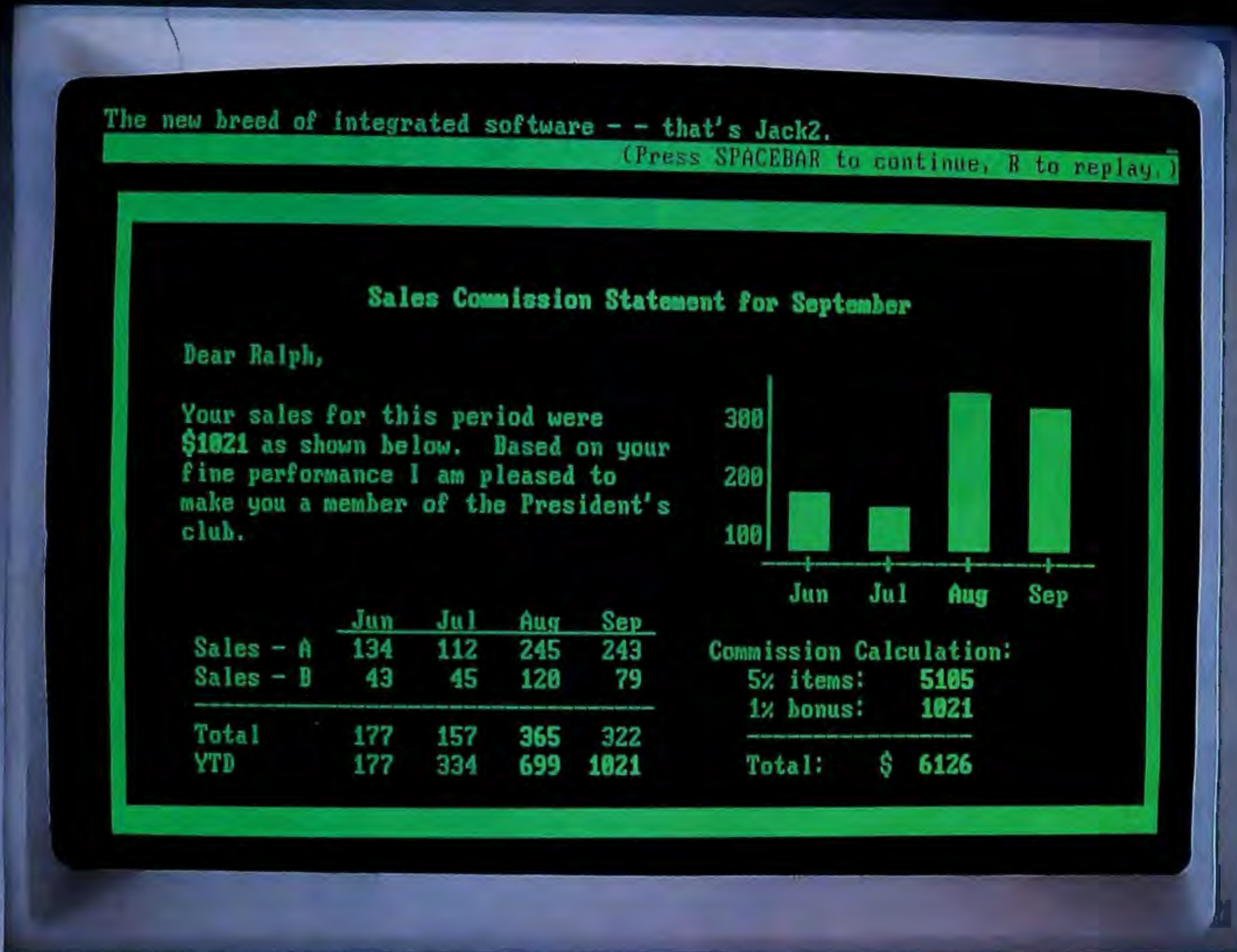
But the real breakthrough behind JACK2 is its ability to edit word processing text, perform spreadsheet calculations, manage data base information and illustrate the results with charts or bars. All at once. All on the same screen.

And while the integrated applications of JACK2 offer unlimited potential as a business tool, individually they offer everything an expert could ask for. Because we've taken the best of all the rest and made them better.

JACK2 offers features other word processing programs only dream of. Like multiple columns of text on the same page that can be edited without affecting other text. Or, an innovative "transparent band" will allow you to view and manipulate confidential notes and calculations in tbe middle of a document, but to your printer they're invisible. Plus, a variable sentence function will tell the computer to watch for a particular word, or spreadsheet comparison, and based on its content, will automatically insert a variety of preselected sentences or phrases in response.

JACK2 spreadsheets allow you to create complex formulas using plain English. Without cell language like A22=C38−G63, your next model will be as easy to remember as Margin=Sales−Cost. You determine individual column widths, depths, justification and decimal points. You can even identify specific cells to only accept telephone, social security, account numbers or other strings.

JACK2 data base management lets you sort records with a choice of three levels. Create additional forms with up to 1000 fields. Design the form format



right on the screen, line by line, column by column. Plus, you can change the original form at any time, adding, deleting or altering fields. To reformat all previous records, simply update them, using the new format without any loss of data.

The beauty of the JACK2 charting function is in its simplicity. What you see on the screen is what we print on the page. There's no need for expensive plotting equipment to produce a presentation quality bar chart or point graph. You determine the titles and values of the x and y axes. You specify the scale. You choose whether headings fall below the line or above individual bars. Or let JACK2 decide.

So, if you've been searching for a new breed of integrated software for the IBM PC and XT, you've found it. From word processiug, to spreadsheets, to data base management and charting, only JACK2 will let you do everything you've always wanted to do. On the same screen. At the same time.

JACK2 is available for the IBM-PC with 128K and two double sided disk drives, the IBM-XT with 128K and one double sided disk drive with supplemental JACK2 XT Installation Package as well as other IBM PC-compatible computers.

Jack2

Business Solutions, Inc.
60 East Main Street, Kings Park, NY 11754 • (516) 269-1120

MICRO FINANCE

by Ken Landis

ProFin

Fortune 1000 corporations have small armies of financial experts who do nothing but analyze the performances of their businesses and look for ways to improve profitability and growth. They may focus their attention on existing operations, or they may be on the lookout for companies to acquire. The techniques these analysts use are not esoteric, magical, or proprietary. Rather, their toolbox contains basic financial instruments, including financial information, financial analysis formulas, and statistical analysis methods, all of which are used in creating the final report.

Financial analysts have used computer systems for a long time. Over the past few years, many of these experts have begun to rely on microcomputers for support; the relatively high cost of timesharing has made microcomputers a financially attractive alternative. Until recently, the application programs that ran on micros were not nearly as sophisticated or powerful as mainframe-based systems. Lately, however, an entire industry has sprouted to supply professional financial analysis programs for microcomputers. This month we'll examine one such program, *ProFin*, a structured financial analysis package from Business Software of Chatsworth, New South Wales, Australia. (The American distributor is Business Software of Los Angeles, California.)

Rather than adhering to our traditional case-study format this time, let's look directly at the software package itself. Since the *ProFin* software requires you to supply a diverse number of inputs in order to set its analysis in motion, your time is probably better spent discussing these inputs, their characteristics, and the way they are used, rather than in focusing on a specific business problem.

ProFin is a menu-driven, structured financial analysis package designed to be used in place of a spreadsheet to perform pro forma financial analysis. *ProFin* is capable of projecting up to thirty-two years into the future and handling up to fifteen products, fifteen expense items, fifteen capital expenditures, and fifteen loans per analysis.

The program's memory requirements are fixed; a minimum of 64K is required. (The formula for determining whether a specific case will fit within *ProFin* is fully documented and discussed in the manual.) Unlike most programs, the amount of memory available is not what restricts the case size in *ProFin*. Instead of looking at the case size and allocating the appropriate amount of memory for it, the program has a fixed maximum case size, which is determined by the number of variables used in the case.

ProFin expects the user to provide production, price and cost data, capital expenditures data (including depreciation and investment tax credit information), equity data, and income tax data. In return, it will provide financial analysis information in the form of an income statement, an interest schedule, a capital expenditure report, a tax schedule, a return on equity and discounted cash-flow analysis, and a balance sheet.

To begin, the user must specify the number of periods to be forecast. Once that's been established, *ProFin* requests the number of products to be considered in the analysis and the production information needed to perform the analysis on each.

The first piece of production information the program requests is the product name for the first product, which may be up to fourteen characters long. Next, the estimated quantity to be produced must be specified.

ProFin has two product-quantity options. The first allows the user to specify a base-year production amount that is increased annually at a specified percentage rate. For example, if you were to estimate that five hundred units would be produced the first year, and that this number would grow at the rate of 10 percent per year, *ProFin* would immediately calculate the quantity of product that would be produced each year and would display the results of its calculations on screen. If any of the projections looked unreasonable, you could change the product quantity number via the editing screen. (*ProFin* can also handle negative growth, or declining production, scenarios.) The second product-quantity option allows the user to specify the exact quantity of product that will be created each year.

ProFin's editing facility can be used to transform data to reflect currency translation or unit measurement changes. It's possible to change a single piece of data or a range of data within a line item, and it's also possible to multiply the existing data by any factor. Say, for example, you were making packaging arrangements for your product and decided to put six units in a case. If you then changed your mind and decided to put twelve units in a case instead, you could use *ProFin*'s editing capability to multiply the number of cases by .5 to arrive at the new case production figure. If this facility were not available, you'd have to change each number manually, which would be time-consuming.

The program now asks whether the user wants the production figures rounded; many people prefer this. Using the software to round the numbers is more effective and efficient than doing it manually; since the program always follows the same rule when rounding, the rounded data it supplies should retain its original consistency.

Once the question of rounding has been settled, the user is prompted to supply the name of the second product. This product data entry sequence is repeated until all the products to be considered in an analysis have been entered, or until the maximum number of products (fifteen) is reached.

Now it's time to enter price data. Selling prices for each product must be entered in terms of the yearly base selling price per unit and will then be compounded by the user-supplied inflation-rate assumption. Throughout *ProFin*, the term "inflation rate" refers to price or cost increase assumptions; many case situations — especially those that are highly price-elastic — will not fully reflect the rate of inflation.

Introducing the Multifunction Boards from Prelude

A Prelude multifunction board is the smartest choice for your IBM PC or PC-XT. You can take advantage of the full potential built into your PC...without wasting future expansion space. Today's high productivity software requires plenty of memory and the input/output capability to print, plot and communicate. For one of the widest selections of IBM compatible memory and I/O options...at sensible prices, consider Prelude...the smart choice.

Memory Options—Add 0 to 384K bytes of parity checked memory to your existing system memory. You can expand memory in 64K increments to the maximum of 640K.

Battery backed-up memory of 0 to 4K bytes will retain data even when your PC is turned off.

Clock Calendar—You no longer need to type in the time and date after powering up your PC. An easily replaceable Lithium battery on the board powers the quartz crystal clock when the PC is off.

Printer Port—The printer port is used to connect a dot matrix printer or other parallel device to your PC. It is configurable as LPT1, LPT2 or LPT3.

Serial Port—Connect a serial device such as a modem, plotter, letter quality printer or even a mouse. The serial port is configurable as COM1 or COM2.

Game Adapter Port—You are missing an exciting use of your computer without this option. It allows you to connect a touch tablet or an IBM-type joystick to your PC.

Software—Each Prelude multifunction board comes with the software to allow you to realize the potential of the added hardware features. You won't have to wait on the printer when you use the Prelude Spool software. Spool does your printing as a background task while you continue with your program. You get software for setting the clock calendar and initializing DOS time and date upon power up. The Appointments Calendar program uses the battery backed-up memory to remember future dates and keep you on schedule. All Prelude sc' are is compatible with DOS 1.1 and 2.0

Prelude Quality—Each board is constructed from the finest quality materials, burned-in and then tested extensively to assure the highest reliability. Necessary cables and complete installation documentation is included.

Warranty—All Prelude products are backed by a one year limited warranty on parts and labor. Optional yearly extensions to the warranty are available.

Call For Prices—As we said, our prices are sensible. We offer substantial volume discounts to corporate buyers, dealers and distributors. (408) 257-6033.

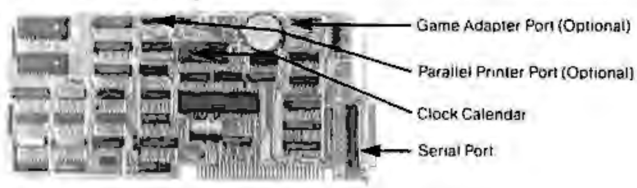
master charge · VISA

20380 Town Center Lane, Suite 217
Cupertino, CA 95014 (408) 257-6033

Six Function Board

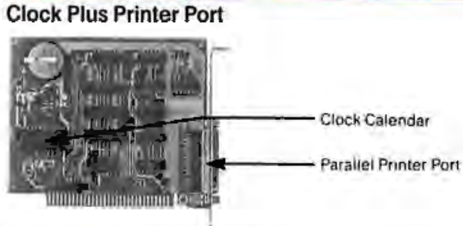
0-256K Parity Checked Memory

I/O Plus NV Memory

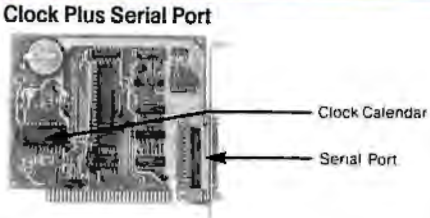


0-4K Memory with Battery Back-Up (Optional)

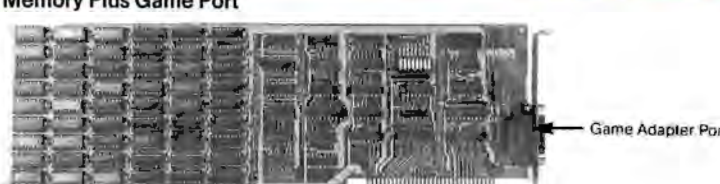
Clock Plus Printer Port



Clock Plus Serial Port



Memory Plus Game Port



0-384K Parity Checked Memory

PCjr—Prelude multifunction products for the PCjr will be available soon.

prelude
Computer Corporation

The next entries to be made have to do with product costs. If you have one product and one cost, the program prompts you to enter a single-cost figure. If you have two products and two costs, you must enter the cost for each product. This arrangement could provide valuable information if these cost-breakdown figures were reflected on *ProFin*'s reports. Unfortunately, they're not; the numbers for each cost item are added together. If you're using aggregate costs, you can "fool" *ProFin* by entering the entire cost in one product category and entering zeros as the prices of the other products in the same category. Because of this cost/product relationship the cost-entry cycle can take an inordinate amount of time.

Costs can be entered on a per-unit basis, calculated as a percentage of revenue, as a percentage of another cost, as an equal fixed cost per year, or as a specific payment each year. The cost-per-unit and fixed-cost-per-year entries can be increased each year by a specified inflation rate.

Once the cost categories have been completely filled in, *ProFin* requests information about the capital expenditures the user expects to incur during the period covered by the analysis. The program assumes that all capital expenditures occur at the beginning of the year. Apparently the authors of this program reasoned that it's better to charge a capital expenditure at the beginning of the year, when it will have the biggest impact on discounted cash-flow analysis, than it is to distribute it evenly through the year or to charge it off at midyear. The program also assumes that all assets are sold at book value in the last year of the analysis. If the user is assuming that the firm is a going concern, then the program's setup could severely skew the analysis. Aware that not everyone will want to make the same assumptions they've made, *ProFin*'s authors have clearly explained how to adjust the data entry or the forecast horizon to compensate for this assumption.

To recognize the sale of a fixed asset, the user must enter a negative capital expenditure in the year the sale is planned. In essence, the sale is used to offset planned expenditures.

ProFin uses two depreciation methods, straight-line and declining balance. For capital expenditure, either method can be used. Declining balance depreciation is limited to three-, five-, ten-, or fifteen-year periods, and the amount depreciated in each year is, according to the documentation, in keeping with federal ACRS regulations. The declining balance percentages *ProFin* uses are set and cannot be changed. These percentages are 25 percent, 38 percent, and 37 percent for three-year capital equipment; 15 percent, 22 percent, and three years of 21 percent for five-year capital equipment; 8 percent, 14 percent, 12 percent, three years at 10 percent, and four years at 9 percent for ten-year capital equipment; and 12 percent, 10 percent, 9 percent, 8 percent, 7 percent, four years at 6 percent, and six years at 5 percent for fifteen-year capital equipment. Investment tax credit information is also entered in the depreciation section of the program.

The investment tax credit (or ITC) can be anywhere between 0 percent and 100 percent. *ProFin* automatically computes the ITC based on the purchase price of the asset and then reduces the asset's taxable basis by 50 percent of the ITC amount. In any one year, *ProFin* allows ITCs of $25,000 plus 85 percent of the balance of the federal tax liability. Unused credits are carried forward to the following year.

Next the program requests equity information. The equity capital amounts the user supplies are used in computing the internal rate of return on capital, effective yield, the net cash flow, and the payback period. *ProFin* excludes equity capital from the discounted cash-flow and net-present-value calculations.

As it does with capital expenditures, *ProFin* assumes that loan balances (or drawdowns, as they are called within the program) occur at the beginning of a year. If repayments do not start in the year in which the obligation occurred (they are deferred), then the interest is capitalized into the loan balance until payments do start. The interest cost, together with depreciation, is deducted from operating income to determine the correct tax liability or operating losses.

If the loan obligation represents a line of credit that is drawn on periodically throughout the analysis, *ProFin* assumes that the entire balance is incurred at the beginning of the initial period and calculates and deducts principal and interest payments based on that assumption. To override this assumption, the user can set up multiple loans (up to a maximum of fifteen), with the loan balances occurring in the period desired.

The repayment options available to the user are blended (self-amortizing), interest-only, fixed principal repayments, and specified principal repayments. For all four forms of repayment, *ProFin* automatically calculates the interest and principal payments for each year. Interest costs are based on the interest rate chosen for each year; this rate may be constant throughout the life of the loan or it may vary from year to year.

For all loans, *ProFin* repays the principal during the last year of the analysis. In a fixed principal loan or a specific repayments loan, the program does not check the relationships among principal payments, the number of years of the loan, and the loan drawdown. The user must ensure that the relationships are correct; otherwise, the resulting analysis may be badly skewed. In either type of loan, *ProFin* calculates the interest cost on the basis of the

outstanding principal balance for the year in question.

Income tax data is the last information the user is asked to supply. A table in *ProFin* lists the tax rate break points and their associated tax rates. The user may change any of these rates and may also supply a state tax rate. Each *ProFin* analysis stores this table with the data on disk, so if you change the federal or state tax rates, only the current analysis will be affected. The user is asked whether the company in question can carry losses forward for state tax purposes.

Now that *ProFin* has product quantity, price, and product information, as well as capital expenditures, equity, and income tax data, it's ready to display or print out its reports. As stated earlier, the program can generate an income statement, an interest schedule, a capital expenditure report, a tax schedule, a return on equity and discounted cash-flow analysis, and a balance sheet.

The reports are easy to read and contain a good deal of useful information. Useful, that is, if you know what you're doing. *ProFin* is a sophisticated financial analysis system. It handles a large number of variables and affords the user a great deal of leeway in entering information and in interpreting it.

As has been said before in this column, many so-called "structured" products are not only structured, but rigid. While structure improves the quality of an analysis, rigidity limits its value. *ProFin* is, without a doubt, a very flexible financial analysis system.

To the best of our knowledge, the package does not break any laws in depreciating equipment, charging off expenses, or paying taxes. (Unlike a package we worked with a few months ago, *ProFin* assumes that the entire tax liability is paid in the year incurred, so rest easy. The IRS won't be coming after you, saying that your program has a lot in common with a convicted embezzler.)

On the other side of the ledger, the program is written in Basic and has not been compiled for faster speed. It doesn't take an extraordinarily long time to run, but in comparison to a compiled or machine language program it is slow.

Surprisingly, the program has no default settings — to say yes or no to a question, you must enter a Y or an N response. This is both good and bad — good because you can't accidentally hit the enter key and go on, bad in that it takes a little while to get used to.

Our tests of the program revealed that it's possible to enter a loan repayment period longer than the period covered by the analysis. The program ignores any period that exceeds the analysis period, but it doesn't tell you that it does this. In other words, if you instruct the program to repay a loan over ten years and the analysis period is only five, it will give the figures for repaying the loan over five years. This is probably the right approach, but it means that if you want to repay a loan over a longer period, you'll need to recompute the numbers yourself.

When you've finished viewing the *ProFin* reports, you can save them in a *Multiplan*-compatible disk file or in a DIF file for use with various other programs. This convenient feature makes it possible to do such things as reading the income statement into a spreadsheet to compute line items as a percentage of gross expenses and calculating the deltas (percentage differences) from year to year.

ProFin is a solid, well-executed, structured financial analysis product. Many of the calculations it performs would be quite difficult for the average user to reproduce using a spreadsheet program. The reports it produces are meaningful and, as best we can determine after fairly rigorous testing, accurate.

At $295, *ProFin* is definitely a product worth considering. Of the structured financial analysis packages we've examined so far, *ProFin* is among the best in terms of price, documentation, performance, and functionality. ▲

KEYS TO SUCCESS

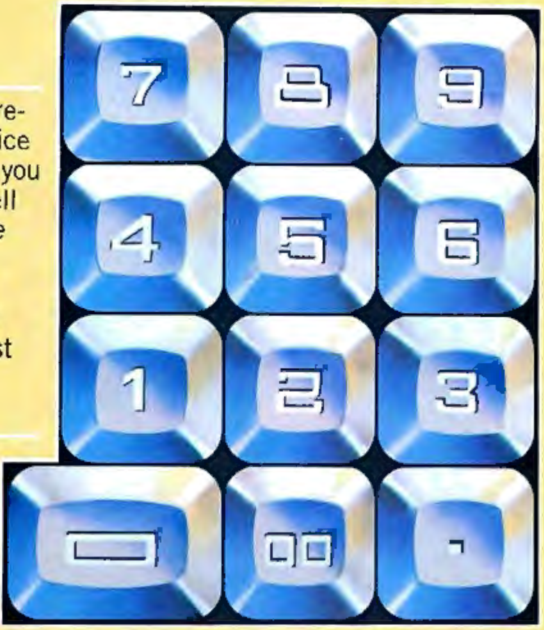
Now available for IBM PC

The keys to success.

Whether you have a small retail company, manage a service agency or run a cattle ranch, you have to do something else well to build a business—*manage money.*

You've got to analyze cash flow, see trends and forecast profitability in a time and cost effective manner.

The right fit.



A computerized accounting system makes sense, but finding the right software is important. The "home checkbook" programs are easy to use but they don't have the advanced reporting capabilities you need. And many small businesses just don't need the complexity or the cost of the modular systems.

Practical Accountant™ is a single entry, small business accounting program that comes complete in one package. It's friendly enough to balance your checkbook, yet sophisticated enough to give you the kind of money management information you need for a successful business.

We could have called it "Accounting Made Easy."

Practical Accountant is designed around fundamental accounting procedures. Even if you don't know much about accounting it is an ideal program.

The manual explains basic accounting terminology and provides a comprehensive tutorial. It will take you step-by-step through the process of setting up a complete, single entry accounting system you can begin using right away.

It's quick to learn and convenient to use.

The program guides you with easy-to-use menus and provides on the screen prompting. Practical Accountant even does some of your work for you like organizing entries in date order—even if you don't.

It's very flexible. You set it up to fit your business.

Practical Accountant allows you to set up your "Chart of Accounts" (income and expense categories) with up to 50 categories, 300 sub-categories and with 20 tax type definitions—all defined by you. The manual shows you how to set up an accounting system for a small business, complete with samples.

Cash Flow Reporting—money management information.

Perhaps the most valuable feature of the program is the cash flow reporting capability. You can look at cash flow by category, by sub-category and by tax type to analyze profitability, tax consequences and general performance. All of your reports are professionally formatted and you have access to profitability information that leads to sound business management.

There is more to this success story.

If you would like to know more about Practical Accountant your dealer can give you a great demonstration and a brochure with a more complete story, or call Softlink TOLL FREE at 800 633-6300 (in CA 800 222-1244) and we will send you the brochure.

Practical Accountant™

Single entry, small business accounting for the Apple IIe, IBM PC and compatibles.

$149.95
suggested retail

SOFTLINK™
3255-2 Scott Blvd.
Santa Clara, CA 95051
408/988-8011

Data in


MicroPro.
insoft
DATA DESIGN
DATA BASE MANAGEMENT SYSTEM FOR THE IBM PERSONAL COMPUTER
WORDSTAR.
123
LOTUS
insoft
Data Base Management System For the IBM Personal Computer
DATA DESIGN
DATA DESIGN
SETUP and UTILITIES
DATA DESIGN
BROWSE and PRINT

Undocumented Keywords in Basic 2.0

by Dan Rollins

BasicA Version 2.0 contains several powerful commands that are not described in the Basic manual. Why would Microsoft include commands and functions in the Basic interpreter but provide no documentation to help you use them?

There could be several reasons, but the most likely is that there were still a few bugs in the code and the folks at IBM and Microsoft decided not to let us know. The new Basic had to be available with the release of the XT, and there just wasn't enough time to groom the code completely.

Pages 3-6 through 3-8 of the Basic manual list the Basic reserved words—the keywords that may not be used as variables. Comparing this list with the reserved-word list of Basic version 1.1, we can see several additions. Figure 1 lists the new reserved words.

Documented	Undocumented
CHDIR	INTER$
MKDIR	KEY$
RMDIR	
	ERDEV
PMAP	ERDEV$
VIEW	
WINDOW	ENVIRON
	ENVIRON$
TIMER	IOCTL
	IOCTL$
	SHELL

Figure 1. New keywords found in Basic 2.00

The statements on the left side of the figure are covered in detail in the Basic manual. *Chdir, mkdir,* and *rmdir* are tools used to keep track of and access the new tree-structured disk directory system. *Pmap, view,* and *window* are new enhancements to BasicA's graphics. The *timer* statement gives Basic a sort of multitasking capability—it allows a subroutine to be executed in the "background" regardless of what the main Basic program is doing.

On the right side of the figure are eight commands that are mentioned in the Basic manual only once—when we are warned not to use them as variable names. The top keyword, *inter$*, does not even belong in the Basic keyword list. It is not tokenized by Basic, nor is it recognized as a keyword. In spite of its inclusion in the list, *inter$* may be used as a variable name. It's possible that a later release of Basic will support this command, perhaps as some sort of substring extraction tool.

Key$* generates a syntax error, as do *pen$ and *string$*; Basic contains no code to check for that trailing dollar sign.

Erdev and *erdev$* are tokenized by Basic, but the jump vector just sends control immediately to the syntax-error handler. These keywords probably are reserved so that a later release of Basic can provide some control over the DOS Standard Error Device—a concept not yet fully implemented, even at the DOS level.

The other five keywords provide an interface for some of the new MS-DOS 2.0 services. These commands are the focus of this article. Two keywords access the system's *environment,* which provides a rudi-

Illustrations by Rick Griffin


ENVIRON

mentary form of communication between programs; two commands are for accessing special device drivers; and one command opens whole new vistas to the Basic programmer—the ability to invoke DOS commands from Basic. Let's look at each.

About the Environment. These two commands provide access to the MS-DOS environment. The environment is a series of ASCII strings (totaling less than 32K) that provides a means of communicating between programs. The environment may be examined and modified by any program. The DOS commands *set*, *path*, and *prompt* all place strings of characters into the environment. A *shell=* command in the Config.sys file also places a string in the environment.

If you are unfamiliar with the environment concept, try a few experiments from the DOS prompt. The DOS *set* command usually is invoked with a parameter and a value (*set* parameter=value). When used without an argument, *set* will show you the current state of the system environment. For instance, the following sequence adds a string to the environment and then checks to make sure that the string was added (the commands typed by you are underlined):

```
A>SET
COMSPEC=A:\COMMAND.COM
A>SET parm=this string
A>SET
COMSPEC=A:\COMMAND.COM
PARM=this string
A>
```

The parameter and the value that was placed in the environment by the second *set* command are displayed by the third one. DOS automatically forces the left side (the parameter) into uppercase. The equals sign is mandatory, and the parameter must be at least one character in length. If the right side of the equals sign (the value) is blank, then the parameter is deleted from the environment.

Notice that the environment always contains a *comspec=* value. This is DOS's way of reminding itself of the name of the program that interprets user commands and executes batch files. As you may know, DOS occasionally needs to reload this program (it's usually Command.com) when another program overwrites part of it.

The DOS environment seems to have little use except in assembly language programs. Assembly language programs can examine all the characters in the environment by finding a pointer passed to it by DOS (address 2CH in the program segment prefix). Programs can spawn a "child process" and pass it new environment information by using the DOS exec service (INT 21H with AH=4BH and AL=0). Thus, an assembly language program can read environment parameters entered by the user and can pass information from program to program. The Basic *environ* and *environ$* commands extend these capabilities to the Basic programmer.

Environ. The *environ* statement places a string into the DOS environment. The format is:

```
ENVIRON string
```

where *string* is a string literal, a string variable, or a string expression, normally in the format

```
parameter=value
```

Here are some examples:

```
ENVIRON "datafile=c:\level1\oct1984.dat"
ENVIRON e$
ENVIRON "datafile="+filename$
```

There are several bugs in this command. First, Basic will accept a command like

```
ENVIRON "datafile myfile.dat"
```

but it will drop the first character of the *value* string as it is placed into the environment, leaving the foregoing as

```
datafile=yfile.txt
```

Unlike the DOS *set* command, the Basic *environ* command does not force any part of the parameter into uppercase. Also, the utility of this command is limited by the fact that strings placed into the environment are *not* passed back to DOS when Basic is terminated via the *system* command (the environment of a "child" is never passed back to its "parent"). Nevertheless, this command is very useful in conjunction with the *shell* command, which we'll come to later.

Finally, Basic limits the size of its environment. During initialization, Basic saves the starting size, rounded up to the next paragraph. Then each addition is first checked against that size to determine if it will fit. If adding the new string (the parameter, the equals sign, and the value) will cause the environment to grow past this limit, then Basic generates an out of memory error. So, you can change the values of existing parameters as long as the new value is shorter than the old or doesn't overflow the up to fifteen bytes of padding that are added to the size of the inherited environment. You can create a larger environment by first using the DOS *set* command to insert a large dummy string before you start Basic, then using the *environ* statement to delete that string before attempting to add new parameters and values.

Environ$. The *environ$* function extracts a *value* from the environment that corresponds to a *parameter* that matches its argument. The format is

```
var$ = ENVIRON$(string)
```

where *string* is any valid string expression. Basic looks through the DOS environment. If it finds a parameter that matches the *string*, then it returns the characters on the right side of the equals sign—that is, the value of the parameter. The string argument must match exactly with that in the environment. Upper and lower case are significant. For instance, if you enter this line from the Basic command mode:

```
PRINT ENVIRON$("COMSPEC")
```

Basic will print

```
A:\COMMAND.COM
```

But "comspec" (lowercase) will not match anything, so Basic will return a null string. A parameter of "COMSPEC " (with a space at the end) generates an illegal function call error. Maybe this was the bug that kept the *environ$* function out of the documentation: DOS allows environment parameters that include spaces, but Basic rejects them. Oddly enough, Basic allows filenames that have embedded spaces (though the *files* command displays them incorrectly), but DOS uses the space as a delimiter, so you can't copy to or from files so named.

What is the (environmental) impact of these two undocumented commands? They provide an easy way to pass information to a Basic program. For instance, imagine a DOS-level batch command that invokes your Basic accounting program. The user might typically need to remember the complicated filename that contains the current month's data. You, the programmer, could automate the sequence. You could place the name of that data file in the environment via a batch command and then have the Basic program find and use that filename. For instance:

```
REM inv.bat ** brings up the INVENTORY program **
SET DATAFILE=oct84i-1.dat
Basic c:invent
•
•
•
10 '** in the Basic program
20 FILENAME$=ENVIRON$("DATAFILE")
30 IF FILENAME$="" then INPUT "What file";FILENAME$
```

You can surely think of extensions to this concept.

The *environ* statement can be used to save some simple strings in one program that must be accessed by another program. This might be superior to using *chain* with *common* for some applications. Because the environment is passed to all DOS programs, the changes made with this Basic command are inherited by any "child" processes created with the *shell* command.

Device Driver Control. There are two apparently functional Basic

commands that were probably left undocumented because there was no way to demonstrate their use. They provide a means to access the DOS 2.0 IOCTL (input/output control) service for certain device drivers.

The concept of IOCTL is a good one: Installable device drivers may be written so that you can talk to the driver without talking to the device. The best example is an asynchronous communications device driver that must occasionally be reinitialized to provide changes in baud-rate, parity, and stop-bit values. The Com1, Com2, and Aux drivers require special coding that changes these parameters from outside the program—that is, code that alters the settings of certain variables. Currently this function is handled by the DOS *mode* command or the Basic *open com. . .* statement.

An alternative would be to send escape characters to the device. Then the driver would need to check each character to see if it is to be passed through to the device or to be interpreted as a control code. This is how the Ansi.sys device driver functions.

There have been several rumors—and a lot of confusion—surrounding the undocumented *shell* command.

The DOS 2.0 installable device drivers provide a convenient alternative. There is a secondary "channel" or pathway through which a program may communicate with a device driver. Using DOS service 44H, a program may send messages to a device. What's more, it also may request information from a device. This provides a flexibility that does not exist with the current device drivers supplied with DOS. For instance, the current printer drivers have no direct way to know if the printer is busy—they must make calls to the ROM printer I/O handler, which is hardware-dependent (it checks a certain bit of a particular input port). But using the DOS IOCTL service, a printer driver could be queried about its status. This would provide an added measure of insulation between a device and the operating system.

You can examine any device driver file to ascertain if it accepts IOCTL requests. Just use Debug to load it and examine the sixth byte (at DS:0105). If bit 6 of this byte is a 1, then the device should handle IOCTL requests. If you have a device driver that accepts IOCTL, you can use the Basic *ioctl* statement and the *ioctl$* function to communicate with it.

Ioctl. The Basic *ioctl* statement passes string data to a device driver that is written to accept IOCTL control strings. The format is:

IOCTL [#]filenum,string

where *filenum* is the number used to identify the device by an *open* statement and *string* is any string literal, string variable, or string expression. The crosshatch is optional.

The *string* argument is passed to the IOCTL channel for the device associated with the *filenum* argument. Some examples:

```
OPEN "plotter" for output as #1     '** plotter device must be
                                        installed
•
•
•
IOCTL #1,"INIT"                     '** send control strings
IOCTL 1,CHR$(27)+CHR$(0)            '** that the driver
                                        expects
```

Unless the indicated device driver is equipped to handle *ioctl* strings, using the statement will cause an illegal function error. A note to assembly language programmers: The Basic code actually invokes the DOS IOCTL service (function 44H, subfunction 3) with CX = the length of the string argument, DS:DX pointing to the start of the string, and BX being the DOS file "handle" that Basic created with the *open* statement. If the device is not set up to handle the *ioctl* call, DOS returns an "invalid function" error code (CF=CY and AL=1).

Ioctl$. The *ioctl$* function reads a string of characters from a device driver. The format is:

var$ = IOCTL$([#]filenum)

As with the *ioctl* statement, the *filenum* parameter must refer to a previously opened device. Otherwise, Basic will generate a bad file number error. Examples:

```
OPEN "plotter" for output as #1     '** plotter device must
                                        be installed
•
•
•
STATUS$ = IOCTL$(#1)                '** read from device
                                        control channel
PRINT IOCTL$(1)
IF IOCTL$(#1)="" THEN . . .
```

This function invokes DOS service 44H, subfunction 2 with DS:DX pointing to Basic's string work area and CX = 255. That is, the driver is requested to return a string of up to 255 characters. If DOS finds that the file handle is for a device that doesn't support IOCTL, Basic generates an illegal function call error. If DOS returns an error code of 13 (invalid data), then Basic generates a device fault error.

These functions will be invaluable for programmers who want to experiment with DOS 2.0 installable device drivers that support IOCTL. When the rest of the world begins writing such drivers, these undocumented Basic commands will be a tremendous asset.

Shell. There have been several rumors—and a lot of confusion—surrounding the undocumented *shell* command. In a nutshell (ahem), this command places at your fingertips all the internal DOS commands and, with one exception, any Exe or Com program. What's more, it allows you to enter the interactive mode of DOS, type a series of commands, and then return to Basic—without losing your place in the Basic program. If this sounds too good to be true. . .well, it almost is.

There is a nasty bug in this Basic statement. Fortunately, the bug can be overcome with a minor "kludge." We'll look at this bug and see how to fix it in a moment. First, here's the format for the *shell* command:

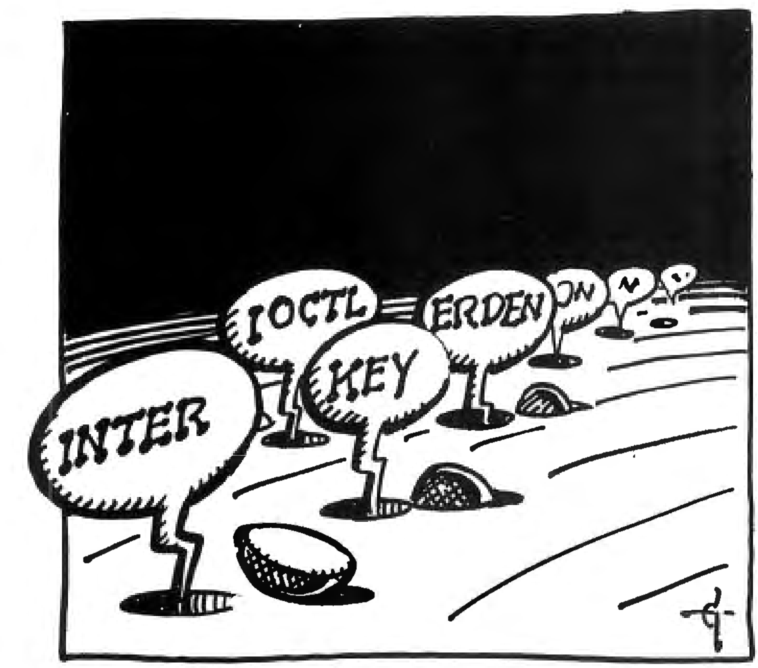
SHELL [string]

where *string* is any string literal, string variable, or string expression. This string is passed to the Comspec program as a command. Thus, in the usual case when the environment indicates that the Comspec program is Command.com, this statement can be used to invoke virtually any service that can be accessed from the DOS prompt.

Some examples:

```
SHELL "dir a:/w/p"
SHELL "mybatch.bat"
SHELL "a:debug a:basic.com"
SHELL "format b:"
SHELL "copy a:file1.txt b:file2.txt"
SHELL "sort <random.txt >sorted.txt"
SHELL "sort <" +tempfile$ +" >" +datafile$
SHELL cmd$
```

Notice in the format that the *string* parameter is optional. When *shell* is used without a parameter, control is passed to the interactive mode of DOS. That is, you temporarily leave Basic and find yourself



The undocumented Basic commands are, at the least, intriguing.

looking at the DOS prompt. At that point you can enter DOS commands at will. To return to Basic, execute the (scantily documented) DOS command, *exit*. Here's a typical sequence:

```
SHELL        [enter a Basic command]
The IBM Personal Computer DOS
Version 2.00 (C)Copyright IBM Corp 1981, 1982, 1983
B>a:chkdsk      [enter DOS commands]
 362496 bytes total disk space
 202752 bytes in 11 user files
 159744 bytes available on disk
 557056 bytes total memory
 163040 bytes free
B>exit
OK           [now you're back in Basic]
```

When you invoke the *shell* command, the Command program is loaded from the path specified by the Comspec parameter of the current environment. That new copy of Command.com is running as a subfunction of Basic. Then Command executes the single statement that was passed to it. If that statement invokes another program (say, Chkdsk), then that program becomes a subfunction of Command.

While *shell* is being executed, there are no less than four complete programs in memory; the first three (Command, Basic, and Command) are just lying dormant. It is perfectly possible to continue building a hierarchy of programs until all your memory is used up. As each program exits, it passes control to the program that invoked it. In this case, when Chkdsk is complete, it hands control back to Command, which returns control to Basic.

Remember that the environment of a child, or secondary, process is a duplicate of that of its parent. Thus, you can use the Basic *environ* statement to pass data to a growing family of programs. Consider that when you are interacting with a Command that is a child of Basic, it *looks* as if you are still at the lower-level Command. To avoid confusion, you can use the *environ* command to change the DOS prompt before invoking Command.com via the *shell* statement. For instance, when you use these commands to run an interactive copy of Command.com:

```
ENVIRON "PROMPT=$n$g$g"
SHELL
```

you will be greeted with a prompt of

```
A>>
```

This, or any other recognizable change to the DOS prompt, will help you identify the level of the currently running Command. When you see this prompt, you will know that the *exit* command will return you to a parent process.

Too Good To Be True? There are several things you will need to know before you begin using the *shell* command in your programs. Your PC will need at least 128K of RAM with about 100K free before you start Basic. Basic must have its full 64K data area plus another 22K for its RAM-resident code area. Then there must be enough room left over—approximately 17K—to load a copy of Command.com. If you want to invoke external as well as internal DOS commands, you will need even more free memory.

Also be aware that Basic and BasicA can never be executed via the *shell* statement. If you execute the Basic command

```
SHELL "Basic myprog.bas"
```

you will see the strange message

You cannot run Basic as a Child of Basic

and you will be returned to the original copy of Basic. The first thing that Basic 2.0 does is check that it is running under DOS 2.0. The next thing it does is check the byte at 0000:050F. If that byte is a 02, then it prints that funny message and returns you to DOS. All we need to do is find a place in the code that accesses the byte at that address.

Well, Basic sets that byte to 02 just before it saves the segment and stack pointer registers and invokes the DOS exec service (function 4BH, subfunction 0). The exec service will load and execute a program—a procedure that makes the invoking function a "parent" and the newly created program a "child." This is what leads to the conclusion that Basic must have a command capable of giving birth to children. If we go through the keyword list, we find the *shell* statement the likely prospect.

Why can Basic spawn programs like Command, Debug, Format, and Chkdsk, but not another copy of itself? The reason is that Basic performs some nonstandard manipulation to low-memory interrupt vectors and data. A second copy of Basic would alter these vectors to point within its own code, thereby corrupting the data for the original copy. This tampering gives Basic the dubious distinction of being nonreentrant. Among other things, Basic tampers with the keyboard

driver. If you have the nerve to override the built-in safety check by changing 0000:050F to a 0 and then executing a baby Basic, you'll soon be reaching for the Big Red Switch.

You can sit at the Basic command mode (the OK prompt) and enter *shell* commands to your heart's content. But when you want to do some serious work by placing *shell* on a program line, you'll run into trouble.

The Bug in the Shell. The serious bug that kept *shell* out of the Basic manual is this: Invoking it destroys a critical pointer in the low part of Basic's data segment. To be specific, it destroys the bytes that tell Basic where your program text begins. The result is that the *shell* command appears to wipe out your entire program, a feat that no one would call user-friendly. The *list* command will display garbage. Even worse, if you try adding a new line of Basic code, you'll enter that twilight zone known as a lockup.

It turns out that there is a relatively simple way to fix this bug. The trick is to save those start-of-text pointer bytes before invoking *shell* and restore them immediately afterward. The relevant bytes are at 30H and 31H in the Basic default segment. Listing 1 demonstrates how to use the *shell* command safely. When you run this program, you will see drive A: start spinning as Command.com is loaded into memory. If the requested DOS command involves executing a DOS external command, then that program must also be loaded. If you want to take advantage of a DOS filter, as, for example, with this line:

Enter any DOS command: DIR A:|FIND /V "i"|SORT|SORT / +9

then your disk will go through additional fits as Find and Sort are loaded and create the temporary files used as pipes. All this takes time and reduces the utility of the service. The more experienced among you will probably see a simple way to speed up the process: Make sure that your default drive is an electronic disk emulator device. DOS pipelines really pump the juice when their temporary files are created in the RAM drive.

```
10 CLS :KEY OFF
20 INPUT "Enter a DOS command: ",CMD$
30 GOSUB 1000
40 LOCATE 25,1 :PRINT "Press any key to continue";
50 WHILE INKEY$="" :WEND :GOTO 10
997 '**
998 '** subroutine executes the DOS command in CMD$
999 '**
1000 T1=PEEK(&H30) :T2=PEEK(&H31)      '** save the pointer
1010 SHELL CMD$                         '** execute COMMAND.COM
1020 POKE &H30,T1 :POKE &H31,T2         '** restore the pointer
1030 RETURN
```

Listing 1.

Increased throughput is achieved if you make sure that Sort and Find both reside on the RAM drive. But even then, Basic still needs to go out to a physical drive so that it can load in Command.com, which is always invoked by the *shell* command. The solution is to copy that program to your RAM drive and then change the value of the Comspec parameter in the DOS environment.

Listing 2 demonstrates this speed optimization. It copies all relevant files to drive D:, which is assumed to be a RAM drive, and it changes the environment so that when the *shell* statement loads Command, it will spin in from the RAM drive. The program sets the DOS default drive to D: and finally invokes the three-pipe DOS filter. The final DOS command is executed in eight seconds. Compare this time with the forty-two seconds required when all the files are on physical drives. By


YOUR PROGRAM HERE

The best software for the IBM Personal Computer. Could it be yours?

Attention, all programmers. Here's a chance to reach the top.

If you've written software that's completed and runs on the IBM Personal Computer, we could be interested in publishing it.

(We also could be interested if it runs on *another* computer. If we select your software, we'll ask you to adapt it to our system.)

But be advised.

Our expectations are great.

Because the software we publish must be good enough to complement IBM Personal Computer hardware. In fact, the more you take advantage of all our hardware capabilities (see the box at right), the more interested in your software we become.

Think about incorporating color graphics into your program, for example.

Use sound. Consider the power of our keyboard and remember to utilize the ten programmable function keys.

In all cases, we're interested in "friendly" software—with emphasis on quality and wide appeal. Programs with the greatest chance of being published must be easy to use, offer a better way to accomplish a task and provide something special to the user.

What kinds of programs? All kinds.

Education. Entertainment. Personal finance. Data management. Self improvement. Games. Communications. And yes, business.

We select programs that will make the IBM Personal Computer an even more useful tool for modern times.

IBM PERSONAL COMPUTER SPECIFICATIONS

User Memory
64K-640K bytes

Microprocessor
16-bit, 8088

Auxiliary Memory
2 optional internal diskette drives, 5¼" 160KB/180KB or 320KB/360KB per diskette

Keyboard
83 keys, 6 ft. cord attaches to system unit
10 function keys
10-key numeric pad

Diagnostics
Power-on self testing
Parity checking

Display Screens
Color or monochrome
High-resolution
80 characters x 25 lines
Upper and lower case

Operating Systems
DOS, UCSD p-System, CP/M-86†

Languages
BASIC, Pascal, FORTRAN, MACRO Assembler, COBOL

Printer
All-points-addressable graphics capability
Bidirectional
80 characters/second
18 character styles
9 x 9 character matrix

Permanent Memory
(ROM) 40K bytes

Color/Graphics
Text mode:
16 colors
256 characters and symbols in ROM
Graphics mode:
4-color resolution: 320h x 200v
Black & white resolution: 640h x 200v
Simultaneous graphics & text capability

Communications
RS-232-C interface
SDLC, Asynchronous, Bisynchronous protocols
Up to 9600 bits per second

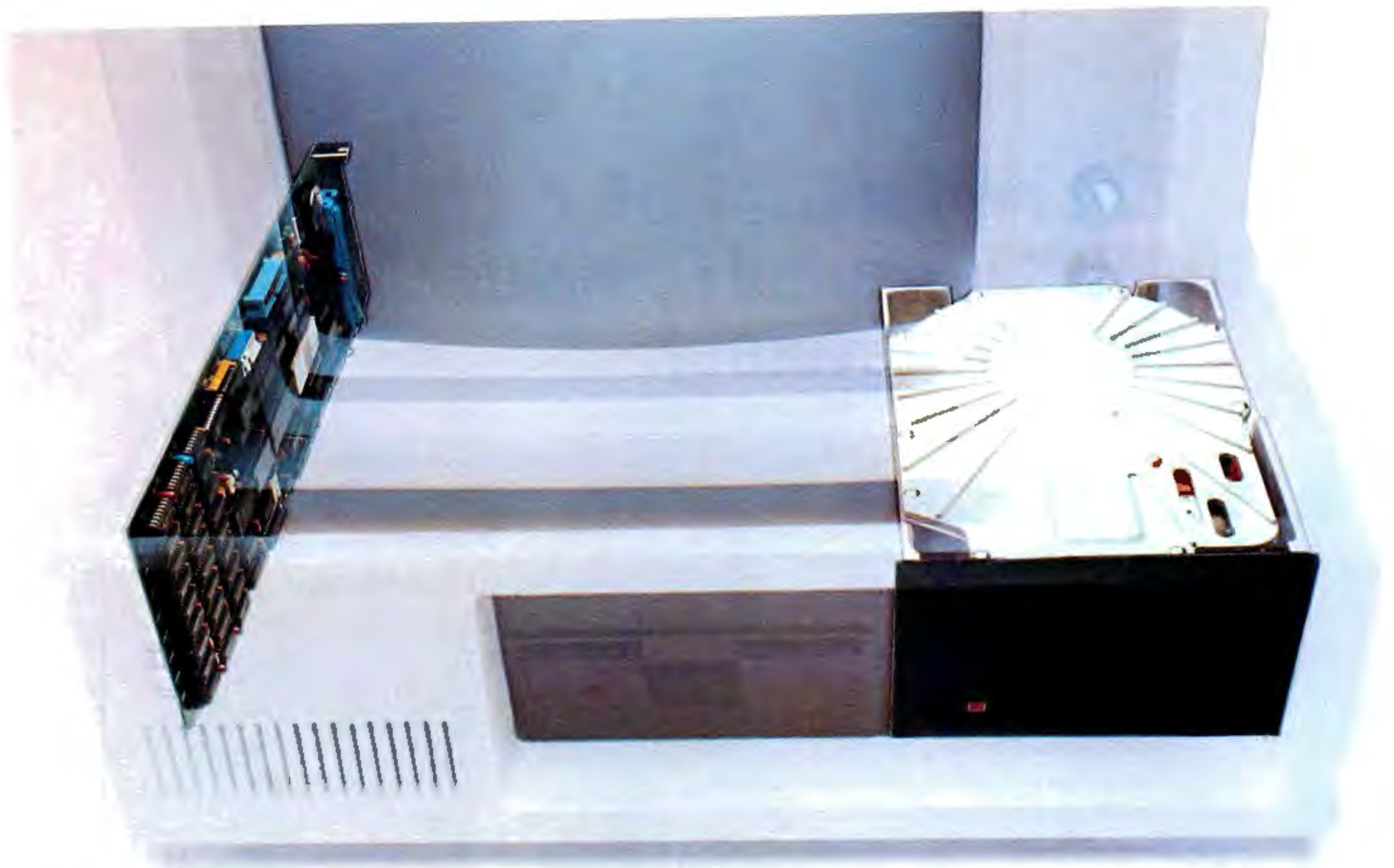
So, if you think your software is the best, consider submitting it. If it's accepted, we'll take care of the publishing, the marketing and the distribution. All you have to do is reap the benefits of our royalty terms. And you're free to market your program elsewhere *at any time* even if you license it to us.

We're offering the ladder. Think about taking the first step.

For information on how to submit your program, write: IBM Personal Computer, External Submissions, Dept. 765 PC, Armonk, New York 10504. IBM®

The IBM Personal Computer
A tool for modern times

For more information on where to buy the IBM Personal Computer, call 800-447-4700. In Alaska or Hawaii, 800-447-0890.
†UCSD p-System is a trademark of the Regents of the University of California. CP/M-86 is a trademark of Digital Research, Inc.



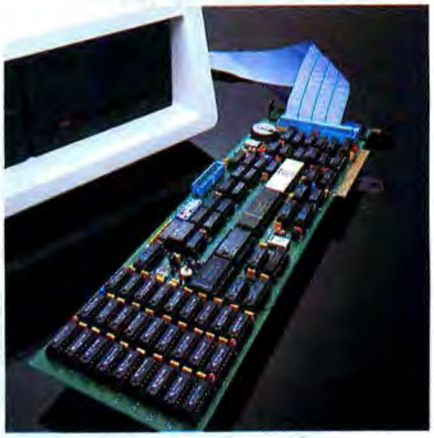
Build a Time Machine

Introducing the first hard disk multi-function system for the IBM PC

Now, you can add 10 or 15 megabytes of on-line hard disk storage to your IBM PC and get a serial port, a clock/calendar, and sockets for as much as 192k of additional system RAM. You get all the features of the IBM XT—and more—in only one expansion slot. It's all possible with Falcon Technology's new PC e**XT**ender™ series hard disk systems.

Falcon PC eXTender Systems Save Both Time and Space

A PC e**XT**ender system will save you time two ways. First, with more on-line capacity than 45 floppy disks, you won't be changing disks continuously. Second, Tim Paterson, the author of the IBM PC's operating system and founder of Falcon Technology, has developed proprietary circuitry that can continuously transfer data at the maximum speed of the drive. The Falcon system is capable of reading one megabyte from the disk in three seconds. This is by far the fastest hard disk available for the PC.



The speed improvements you get with Falcon will turn your PC into a true time *saving* machine.

A PC e**XT**ender system also saves you space. The disk controller and the added functions occupy only one expansion slot.

Plated Media for High Data Density

Falcon's PC e**XT**ender systems use thin-film plated media which resist damage from contaminants and head crashes better than the oxide-coated media used in other PC hard disk systems. In addition, the disk/head cavity is shock mounted to a cast aluminum frame.

Stretch Your Capabilities, Not Your Budget

With a PC e**XT**ender, you'll be able to process more data, faster, and at a lower cost than with an IBM XT.

Prices for Falcon PC e**XT**ender systems start at $2,295.00 for 10MB mounted within the IBM chassis. The 15MB drive is $2,595.00. PC e**XT**ender systems are also available in externally mounted cases. Systems can be configured with one or two Falcon drives.

For the serious PC user, a Falcon PC e**XT**ender may well be the ultimate performance-enhancing add-on available.

To find out more about Falcon PC e**XT**ender systems for your IBM, give us a call:

(206) 251-8282

Dealer inquiries are invited.

Falcon Technology, Inc.
6644 South 196th Street, Suite T-101
Kent, WA 98032

FALCON

```
7 '** Display a directory sorted with a primary key of the
8 '** file extension and a secondary key of the filename.
9 '**
10 CMD$="copy a:command.com d:"       :GOSUB 1000
20 ENVIRON "COMSPEC=D:\COMMAND.COM"              '** so COMMAND will load from D:
30 CMD$="D:"                        :GOSUB 1000  '** make RAM-drive the default
40 CMD$="copy a:sort.exe"           :GOSUB 1000  '** place utilities on RAM-drive
50 CMD$="copy a:find.exe"           :GOSUB 1000
60 Q$=CHR$(34)                                   '** need quotes for FIND filter
70 CMD$="dir a:|find /v"+Q$+"i"+Q$+"|sort|sort /+9"
80 GOSUB 1000 :END
997 '**
998 '** subroutine executes the DOS command in CMD$
999 '**
1000 T1=PEEK(&H30) :T2=PEEK(&H31)                '** save the pointer
1010 SHELL CMD$                                  '** execute COMMAND.COM
1020 POKE &H30,T1 :POKE &H31,T2                  '** restore the pointer
1030 RETURN
```

Listing 2.

using the machine language speed of the Sort filter, you can gain a great advantage over a sluggish Basic string sort.

Normally, when you invoke filters and other programs, control is first routed to Command, which then passes control to the desired program. You might find it convenient to bypass this middleman. Although there is some danger involved, you can use the *environ* statement to change the value of the *comspec=* parameter to a different path and program before you issue the *shell* command. This passes control directly to that program. However, Basic supplies no means to pass parameters to this child process, nor is there any way to redirect the standard input or output. The following Basic statements will load and execute Debug without first invoking Command.

```
ENVIRON "COMSPEC=a:debug.com"
SHELL
```

One final warning about *shell*: The kludge that masks the damage created by the command does not fix the underlying problem. The problem is that DOS thinks that there is a program segment prefix (PSP) at the start of Basic's data segment. It believes that the bytes it modifies are "reserved bytes" in that PSP—critical memory lock control bytes. You can use this head-in-the-sand approach successfully except in one instance. After making several modifications to the DOS environment by invoking the *set* command from Basic, you might run into something like this:

```
Can't Continue after SHELL
Memory allocation error
Cannot load COMMAND, system halted
```

This requires a control-alt-delete reset of the system.

The undocumented Basic commands are, at the least, intriguing. Although the *iotcl* statement and the *ioctl$* function are no good without drivers to use them with, they will one day be very valuable for certain applications. The *environ* statement and *environ$* function are good ideas and handy tools. They may be put right to work to help automate tedious and complex sequences.

With proper care and enough imagination, you can make the *shell* command into a valuable tool. It can make fast work of sorting text files; it can simplify complex, even dangerous commands like *format* and *backup* by providing a means for a program to prompt a novice user through a sequence of steps; it can be used to customize DOS by allowing a programmer to write menu programs and simplified command syntax. In short, it can add speed, flexibility, and user-friendliness to the PC-DOS system. ▲

Quality you expect, at a price you don't.

BECK DOUBLE DENSITY DISKETTES

SINGLE SIDED **$2.19 ea.** / **$2.79 ea.** DOUBLE SIDED

Our message to you is simple. If you like the quality of Dysan, Verbatim, 3M, et al, you'll like the quality of Beck soft sector, 5¼″ flexible diskettes. The only major difference is cost. We're less expensive. In fact, a lot less expensive.

Why does Beck cost less?

Our philosophy is: Excellent quality and reliability, at a cost that beats the jackets off other diskettes. We can do it because we (1) put our money into the product, not mega-marketing schemes and fancy packaging; and (2) sell our money-saving 25-diskette pack to you direct via a toll free order line, so you get fast, door-to-door service efficiently.

When you buy Beck, you've got the best. Beck Quality. Beck Reliability. And, of course, Beck Price.

1D, soft sector 5¼″ diskette $2.19 each
2D, soft sector 5¼″ diskette $2.79 each

For IBM, Apple, TRS and 97% of popular microcomputers.

What about quality and reliability?

At Beck, our success as a diskette manufacturer depends upon our ability to provide you with a fully reliable, quality diskette – every time. For that reason we take no shortcuts. You get the best because we are committed to excellence. Every diskette is manufactured to very strict quality standards. We test and retest 21 times throughout the manufacturing process to insure compliance with no less than 42 rigid specifications. We make sure you get the very best – a 100% certified, 100% error free diskette.

Our satisfaction money-back guarantee and full 7 year warranty are proof of our commitment to excellence and confidence in our product.

(in New Hampshire call 924-3821) ***Door to Door in 48 hrs.***

Order Now Toll Free

VISA MasterCard *COD'S CASH ONLY*

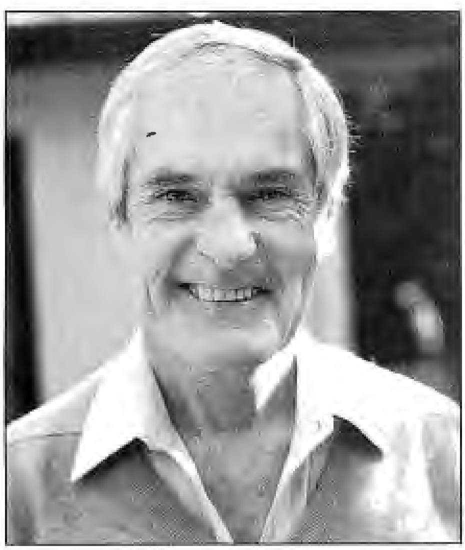
Corporate Accounts Welcome

1-800-BECK MFG

Order Toll Free 1-800-232-5634. Available in 25 pack only, plus freight. Complete with hub reinforcing rings, Tyvek envelopes, color coded user labels, and nonmetallic write protect tabs. All Beck Diskettes meet or exceed ANSI specifications.

tradetalk

Δ Former Harvard faculty member and well-known sixties figure **Timothy Leary** has announced the beginning of a collaboration with **Xor** (Minneapolis, MN). Says Leary, "The project will result in a software product for personal computers as well as a book," which will be in the form of a high-tech novel. The software will be designed to relate closely to the story. "I plan to call it *The Brain Game*, since the point of the product will be intelligence increase in an entertaining format."



Timothy Leary will be designing the Brain Game *for Xor Corporation.*

Having recently acquired a PC, Leary hopes that his software will allow the purchaser to enter into "a symbiotic relationship between a person and a computer"—an altered state more and more computer users are finding themselves in these days.

Δ A new senior product manager for IBM products, **Mary Lou D'Altilia**, has been appointed at **Interlan** (Westford, MA). She will oversee the design, manufacture, and marketing of products compatible with IBM Systems Network Architecture. "D'Altilia brings over nine years of experience in planning and providing specialized technical marketing support for communications and networking products," said Paul J. Severino, company president. Most recently she was a senior product specialist with Wang Laboratories.

Δ The former Microsoft national sales chief, **Nicholas D. Roche**, has been named vice president of commercial sales at **Perfect Software** (Berkeley, CA). In the newly created position, Roche is responsible for all phases of OEM sales and marketing as well as national accounts. Roche has also held management positions with Verbatim and Memorex. In addition, he brings consumer marketing expertise from Procter & Gamble.

Δ Seeking to increase sales in the lucrative insurance market, **IBM** (Armonk, NY) has initiated a private label deal with **Metamorphics** (Bala Cynwyd, PA) to manufacture an insurance package to be sold by IBM. Before the agreement, Metamorphics sold its *Insurance Agency System* through direct mail and insurance industry trade journals. This strategy is the same one IBM used to enter the insurance market with its Series/1 minicomputer.

Δ The game company **Funtastic** (Audobon, PA) has moved again. New digs are 724 Meadowlark Road, Audobon, PA 19403.

Δ **Ashton-Tate** (Culver City, CA) announces the appointment of **Charles Babbit** to the post of executive vice president and chief operating officer. Babbit will oversee the company's operations division, which includes production, the physical plant, the accounting section, and legal services.

Δ **Informatics** (New York, NY) has selected a new vice president of development and support. **Geoffrey Barnes**, promoted from director of development and support, will coordinate the development and conversion of the company's application software internally and with outside contractors. Barnes also serves on the board of New York's Opera Camerata and is coprincipal bassoonist for the New Amsterdam Symphony Orchestra.

Δ **Susan M. Goldberg** was recently appointed vice president of **Micro Lab** (Chicago, IL). Goldberg will be responsible for liaisons with advertising, public relations, and marketing in addition to special projects. Δ Micro Lab has also announced the formation of **Micro Home**, a division for products designed for home use, which joins Micro Fun, Micro Learn, and Micro Lab as divisions of the company.

Δ **PC Technologies** of Atlanta, Georgia (formerly Easitech) has settled out of court in the Quadram/Easitech suit and countersuit. PC Technologies's president, Gary Matte, explained: "Our decision to settle was based on four factors: Quadram was willing to give us a general release precluding future harassment, legal expenses were draining our working capital, our hardware design was based on three-year-old technology and was not competitive, and our marketing and distribution strategies are widely accepted and that is where the true value of the company is." After paying $9,850 to Quadram, the company is now able to proceed with three new expansion boards, currently in the design stage.

Δ **Cullinet Software** (Westwood, MA) has announced the opening of district and regional education centers in thirteen cities across the United States and Canada. The new education centers expand and extend geographically available product training opportunities. Courses offered at the centers emphasize a how-to approach and role training for applications of the company's software, aimed at sys-

tems analysts, systems programmers, and other data processing professionals. Thirty thousand customers are expected to enroll in classes this year.

Δ A former executive with now-defunct Osborne Computer, **James F. Ottinger**, has been named president and chief operating officer at **Forte Data Systems** (Santa Clara, CA). Δ Forte has also appointed two new sales managers: **Jay Anderson** , national sales manager, and **Ed Boswell**, western sales manager. Most recently, Anderson was vice president of Metro Systems. Boswell has directed national accounts sales for Micromation and Wang.

Δ **Kathleen A. Foster** has been named controller at **CompuPro** (Hayward, CA). In this newly created position, she will be responsible for the company's day-to-day accounting functions. Prior to joining the company, Foster was assistant to the controller at the Oakland *Tribune*.

Δ The **Comdex** expositions will extend to Asia in spring 1985 with the premiere of Comdex/Japan, according to **The Interface Group** (Needham, MA), sponsors of the shows. The company now offers three domestic Comdex trade shows, plus Comdex/Europe held annually in the Netherlands. The new Comdex/Winter will premiere at the Los Angeles Convention Center, April 5 through 7, 1984.

Δ **Simon & Schuster** (New York, NY) has named **Frank E. Schwartz** president of the newly formed **Electronic Publishing Division**. S&S pioneered the field of mass market books with the introduction of Pocket Books in 1939. Twenty personal-computer-related titles are scheduled for spring release. Schwartz comes to the company from Wells, Rich, Green Advertising. "With his background and experience, Schwartz is an expert in a field too new to have produced many," according to Richard E. Snyder, S&S chairman and chief executive officer. Δ **Alvin B. Reuben** has been appointed executive vice president, sales and distribution, for the new division. Reuben was most recently president of the company's promotional book publishing division. Δ The entire line of educational software from the **Learning Company** (Portola Valley, CA) will be distributed to book and toy stores by the division. According to Reuben, "We already know from extensive research that mothers are an important purchasing influence in this market and that they are comfortable buying in bookstores." Δ **James E. Korenthal** has been named vice president, technical services. He joins the company from Atari.

Δ **Cdex** (Los Altos, CA) has signed a major agreement with **Deltak Microsystems** (Naperville, IL) to distribute Deltak's line of training courseware, giving Cdex twenty-six computer-based training products. Cdex currently sells through over two thousand computer retailers worldwide.

Δ The establishment of an eastern sales division will be the responsibility of **Brigitte Miklaszewski**, manager of the new eastern regional office of **Schuchardt Software Systems** (San Rafael, CA). Working out of Greenwich, Connecticut, she will supervise all sales, support, and personnel activity in the nineteen eastern states. Prior to joining the company, Miklaszewski held several management positions with Exxon Office Systems. Δ **Larry Bagby**, 35, has recently been named manager of the newly established central regional office located in Dallas, Texas. His responsibilities will be to establish the central sales division and supervise sales, support, and personnel activity in the central states. Prior to joining the company, Bagby was system products specialist for Exxon Office Systems.

Δ **Lotus Development** (Cambridge, MA) has announced the promotion of **Dale Troppito** to vice president of software development. She will be responsible for development of all new products and for support of technical development. Troppito, 31, joined Lotus after a stint with Software Arts. She brings nearly ten years of experience to her new job. Δ **Carol Kochmann** has been promoted to director of finance, having joined the company in 1982 as controller. In her new position, Kochmann will be responsible for bank and investor relations, cash management, and international finance relations.

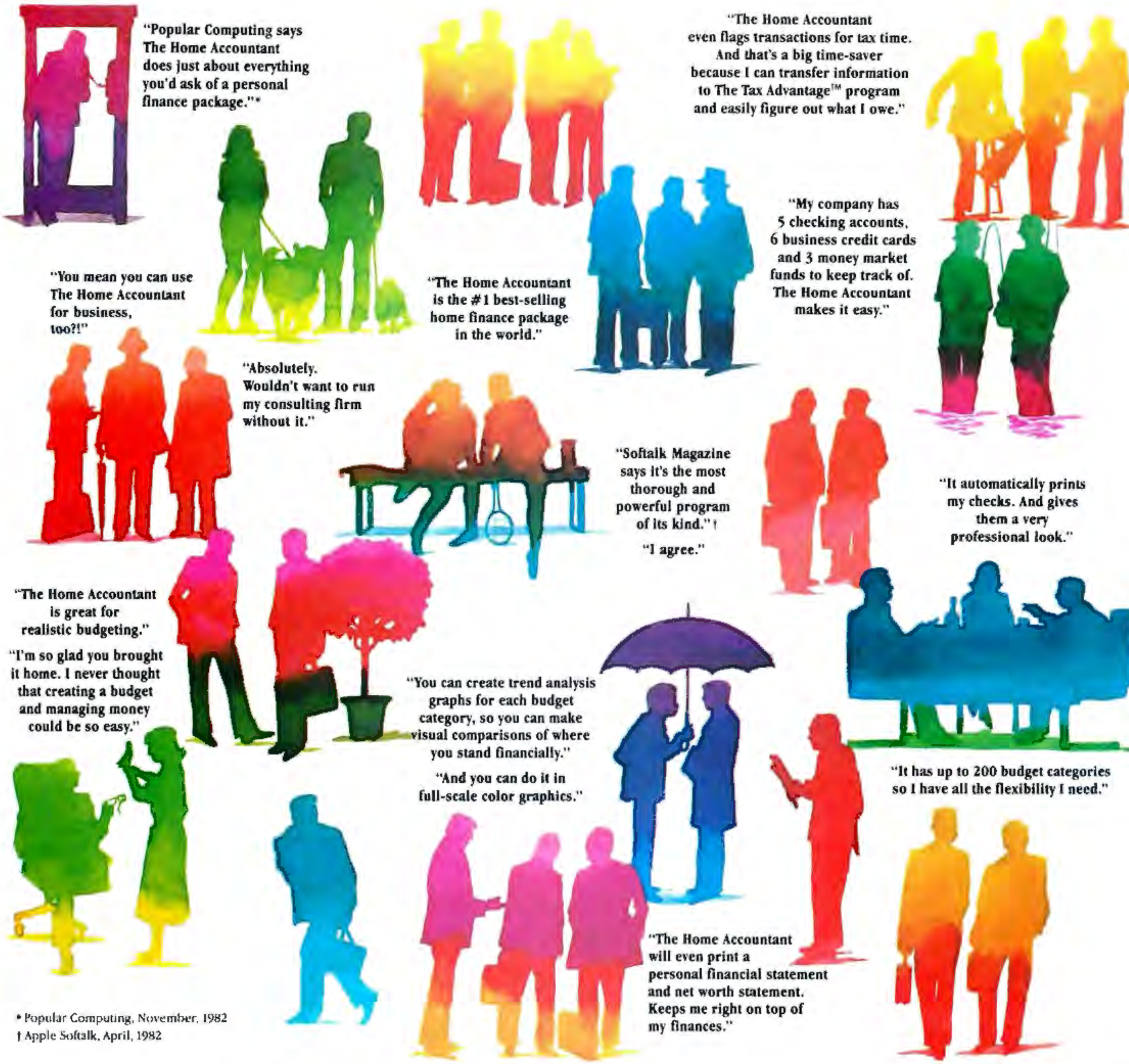
Δ **Jeffrey A. Weber** has been named new corporate controller. He came to Lotus after seven years at Polaroid, where he most recently served as senior financial manager. He will be responsible for planning, designing, and implementing internal financial systems, controls, and procedures. Δ A veteran of IBM, **Steve J. Crummey** has been appointed national retail sales manager. His duties will be to oversee the retail sales operation of authorized Lotus dealers and the distribution of the company's products in the United States and Canada. Crummey spent seventeen years with IBM, most recently as New England regional manager for office and small systems. Δ **Jim Manzi** has been promoted to vice president of marketing and sales. As director of marketing, Manzi had overseen expansion of the company's marketing channels. In his new role, he will continue to develop marketing avenues. Manzi is responsible for dealer and end-user training, marketing communications, and expansion of Lotus activities into international markets.

Δ Second-round financing, garnering $3 million, has been completed by **Concentric Data Systems** (Westboro, MA), which has finalized an agreement with six major venture capital firms. **John Henderson**, company founder and president, said the money will be used to help finance marketing activities—including a comprehensive, seven-figure dealer support program—and to undertake further developmental work. Lead investor in the deal was **The Charles River Partnership** of Boston.

Δ **Software Arts** (Wellesley, MA) has announced the appointment of **Dr. Milos V. Konopasek** as senior scientist. Dr. Konopasek will lead the development of the company's equation-processing software. As a consultant, he originally proposed the groundwork for the *TK!Solver* program. Daniel S. Bricklin, chairman of the company, said, "Milos's pioneering work in the design of very high-level languages and interactive software has been instrumental in the development of the new equation-processing concept. His expertise in the broad areas of applied mathematics, engineering sciences, and operations research will further the company's commitment to meeting the computer needs of noncomputer professionals." Dr. Konopasek is a native of Czechoslovakia.

Δ **Counterpoint Software** (Minneapolis, MN) has moved to larger quarters. Their new address is 4005 West Sixty-fifth Street, Minneapolis, MN 55435. New phone: 612-926-7888.

Δ ***Educational Computer*** magazine (Santa Clara, CA) has also relocated to larger accommodations. New address is 3199 De La Cruz Boulevard, Santa Clara, CA 95050. ▲



* Popular Computing, November, 1982
† Apple Softalk, April, 1982

Everyone's talking about The Home Accountant.™

Is it because it's the #1 bestselling home finance package in the world? Or because it's extremely thorough and powerful and easy to use? Or because it's great for home and business use? Or because it has up to 200 budget categories and handles up to 5 checking accounts?

Yes. But there are a lot more reasons why people buy The Home Accountant. And why you will, too.

Because The Home Accountant can literally save you hours of time. And take the headache out of handling your finances. Whether it's setting up a budget, cataloging your expenses, balancing your checkbooks or handling your credit cards and money market funds. For personal or business use.

The Home Accountant will even print net worth and financial statements. Not to mention being a lifesaver at tax time. Especially when you're able to transfer information onto Continental's The Tax Advantage™ program and figure out what you owe. Quickly.

In short, The Home Accountant is the most effective software program there is for managing your money. And managing it easily.

Stop by your Continental Software dealer today and pick up The Home Accountant. You'll see what everyone's talking about.

The Home Accountant is available for Apple II/IIe, IBM PC/XT, Atari 400/800/1200XL, Osborne,® TRS-80 Models III/4, Commodore 64, Texas Instruments Professional, Zenith Z-100/110, Compaq and KayPro computers. Actual budget capacities will vary with each computer.

For your free 48 page booklet, "Tips For Buying Software," please write Continental Software, 11223 South Hindry Avenue, Los Angeles, CA 90045, 213/417-8031, 213/417-3003.

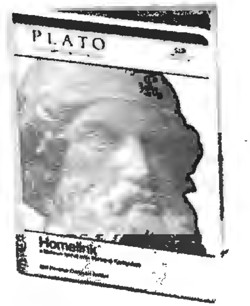
Continental
Software
A Division of Arrays, Inc.

The Home Accountant and The Tax Advantage are registered trademarks of Continental Software.


PLATO Homelink Network
Sign up now for just... $15*
Regular price $50
All the software and instructions you need to use the amazing PLATO Homelink network, in a crush-proof package specially designed to fit next to your IBM PC. Look at what you get.
5¼" Flexible Disk lets you access the PLATO Homelink network from your IBM PC. Completely interactive. Just follow the instructions the software gives you–and you're on!
PLATO
Homelink
Text Processing
Homelink Services
ELECTRONIC MAIL tells how to use PLATO Homelink's electronic mail easily and quickly.
Design graphics from simple to complex with PLATO Homelink and the GRAPHICS DESIGN guide.
ROAD MAP overview booklet explains the features that are available on PLATO Homelink.
How to use the network's word processing capabilities is all here, in TEXT PROCESSING.
ACCESSING PLATO HOMELINK. Detailed "how-to" booklet with information on access, billing, hours of operation, on-line support.
*Introductory offer expires March 31, 1984.

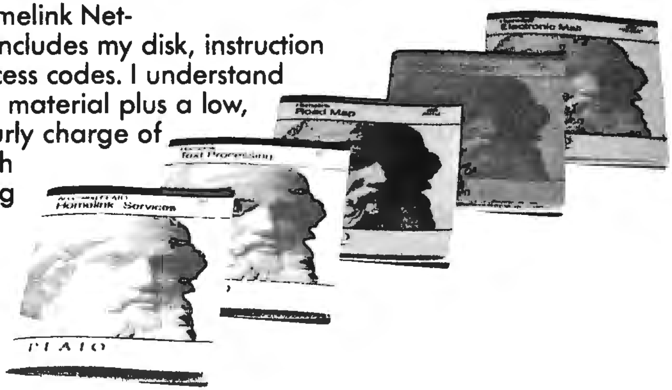
Special Introductory Offer–Only $15.
Expires March 31, 1984*

PLATO® HOMELINK™ NETWORK–A REAL VALUE AT $5 AN HOUR.



☐ **YES**—here's my application for joining the PLATO Homelink network.

Please send me the PLATO Homelink Network Service package, which includes my disk, instruction manuals, and my personal access codes. I understand that I will be billed $15 for this material plus a low, annual fee of $10, plus an hourly charge of $5 for network time used. (With no monthly minimum.) All billing will be by credit card.**



*Regular price $50.

**An additional $2 will be charged for shipping and handling. California residents will be charged an additional 6% sales tax. Additional charge for optional file space is 17¢ per file part per day.

Check to make sure you have the equipment needed for PLATO HOMELINK.

- IBM PC with 64K RAM
- 1 Disk Drive
- Color/Graphics Monitor Adapter
- Asynchronous Communications Adapter (RS232 interface)
- 300 or 1200 baud Modem
- Monitor that operates with the color/graphics adapter.

SERVICE AGREEMENT. I understand this product is marketed subject to a license which limits my use of the product. I will obtain the right to use the PLATO Homelink Network Service by paying the required fees and by agreeing to the terms and conditions in the package. I will be responsible for protecting the security of my personal access codes. A copy of the license, along with the warranty, is available free upon request from Control Data Publishing Co., 4455 Eastgate Mall, San Diego, CA 92121.

Name (Last) (First) (Initial)

Address (Number and Street)

City State Zip

( ) ______________________ Allow 3-5 weeks for delivery.

Telephone

Check one—Modem rate: ☐ 300 baud ☐ 1200 baud

Please bill the fees for the PLATO Homelink Network Service to my (**check one**).

☐ Visa ☐ MasterCard ☐ American Express ☐ Diners

Card Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date________

I will comply with the terms of the license agreement and the operating rules of the service.

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 200108 SAN DIEGO, CA

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA PUBLISHING
P.O. Box 261127
San Diego, California 92126

CORONA

The mainframe world has long provided a comfortable niche for manufacturers who produce computers that run IBM software but offer better price/performance ratios. The quick acceptance of the PC in the microcomputer world and the rapid development of software to run on it and hardware to enhance it have led a growing number of companies to produce computers that are, to some degree, compatible with the PC.

One of the best of these machines is the Corona PC from Corona Data Systems. The Corona PC is available in desktop and portable models that are well designed, well constructed, durable, and

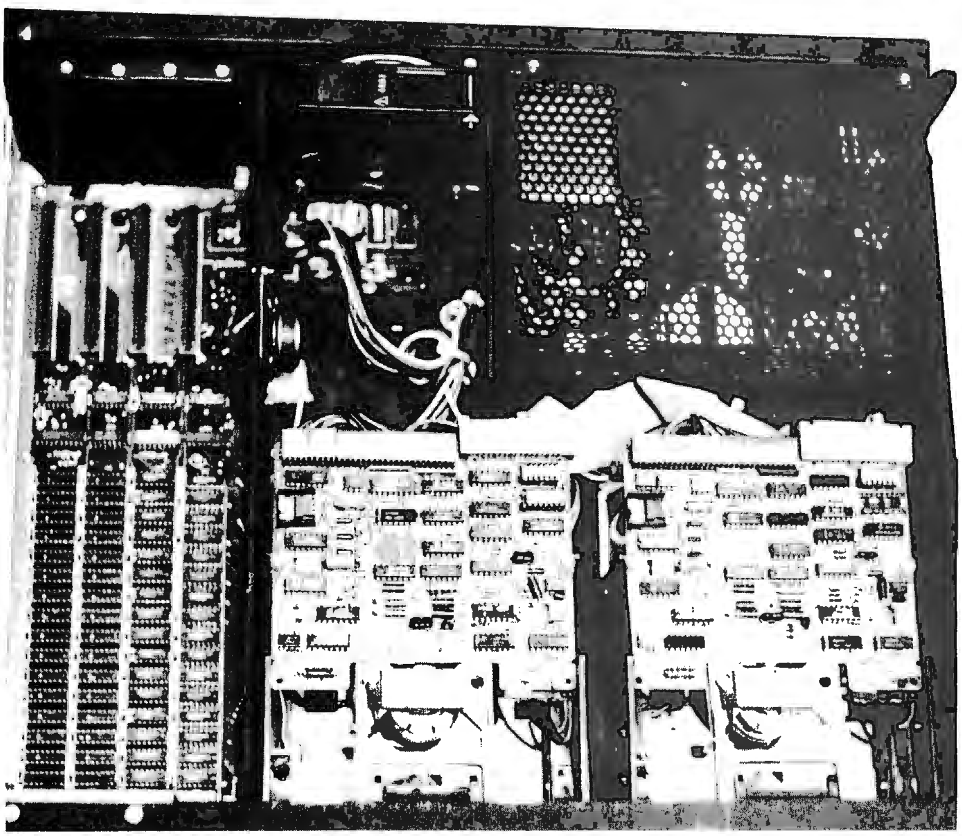
reliable. Corona's desktop PC can be configured with either one or two floppies, or a single floppy and a ten-megabyte hard disk, while the portable is available with either one or two floppy disk drives. All the Corona floppy disk drives are double-sided and double-density.

Both the desktop and portable units have four IBM-compatible expansion slots. In their standard configurations, both machines have 128K of RAM installed on a motherboard that has sockets for 512K. All units are supplied with a built-in display adapter that provides both monochrome and graphic output, as well as built-in serial and parallel ports. Each is sold in a bundle with a high-resolution monochrome display, freestanding for the desktop version and built in for the portable. System and application software, including MS-DOS, GW Basic, PC-Tutor, and the Wang-like word processing package MultiMate, is included in the basic purchase price of all the Corona systems.

A desktop unit equipped with a single

A Review of the Desktop and Portable PC-Compatibles from Corona Data Systems

by Joel Pitt



The Corona desktop PC and portable PC have four expansion slots instead of the PC's five, but more functionality is built into their motherboards than on the PC. Memory (not parity-checked) is organized in four 128K rows, so the Corona user has twice as much onboard capacity as the PC user. The built-in display adapter, which does not consume an expansion slot, provides both monochrome and color graphic output, and the machines offer built-in parallel and serial ports as well. Pictured above is the Corona desktop PC.

floppy disk drive and a ten-megabyte hard disk drive was made available to Softalk *and tested over the course of several months. The following account is an evaluation of this machine. Whatever we say here about compatibility between the desktop Corona and the PC may be assumed to apply to the portable Corona as well.*

Although the Corona units are worthy in their own right and provide some features that aren't ordinarily available on the PC, the Coronas are clearly designed to be marketed primarily on the basis of their PC compatibility. As is evident by now, the word *compatibility* is an elastic term that has been used to describe computers that differ from the PC in many important respects. Copyright and patent laws guarantee that no computer whose front panel is not emblazoned with those three important letters can be identical to IBM's machine, and even the most industrious compatibility seeker

is likely to fail on a point or two.

Some manufacturers claim that their machines are PC compatible merely because the machines, like the PC, use an Intel 8088. Others claim compatibility on the basis of MS-DOS, even if they use incompatible disk formats or sizes. It's reasonable, however, to expect a truly compatible computer to run the vast majority of off-the-shelf IBM software and operating systems, and to provide expansion slots that will accept off-the-shelf hardware add-ons.

Corona's claim to PC compatibility starts, of course, with its use of the 8088, and MS-DOS, and identical disk formats. In addition, Corona's designers have gone to great lengths to ensure that their computer is (virtually) functionally identical to a PC, down to the engineering of its expansion slots and the Basic calls to its ROM BIOS. Because of this, the Corona accepts both off-the-shelf hardware and a variety of off-the-shelf PC software. With minor exceptions, the Corona is an outstanding performer.

The Corona computers are supplied with their maker's own version of MS-DOS. As a result, any software package written to run under either MS-DOS or PC-DOS that conducts its interaction with the hardware using only standard operating system calls runs without any problems on the Corona—as you'd expect. To optimize performance, many PC packages bypass the standard DOS calls and call specific ROM routines directly. This practice is particularly common among word processing packages. The Corona was tested with a variety of word processors—*WordStar, Easy-Writer 1.1, Volkswriter, PC-Write*, and the *MultiMate* program included with the Corona package. All worked perfectly.

Corona's built-in graphics-display processor is incompatible with standard PC graphics. However, a standard PC color/graphics adapter can be mounted in one of the slots. Installing this card is easy, and both it and the Corona work perfectly with a variety of PC graphics packages. For example, the Corona produced business graphics using *PFS:Graph* and *1-2-3*. Graphics-intensive games like the *Microsoft Flight Simulator* and *Olympic Decathlon*, which don't run on many so-called PC compatibles, will run on the Corona.

The list of software tested and successfully run on the Corona includes such products as *1-2-3, VisiCalc, SuperCalc, Multiplan, TK!Solver, Knowledge Man*, and *MicroRIM*. A variety of system tools—such as the *Norton Utilities*, IBM's *Professional* and *Personal Editors*, IBM's Fortran and Pascal compilers, *PC-Forth*, Logitech's Modula-2 compiler, and Lattice C all perform as well on the Corona as they do on the IBM. Even IBM's own DOS 1.1 and 2.0 boot directly on the Corona.

The Basic interpreter supplied with the Corona is one of the few areas of minor incompatibility between the Corona and the PC. Although Corona's GW Basic comes from Microsoft, which also makes the IBM Basics, there are major internal differences, and some external differences, in the ways that Basic is implemented on the two machines.

The internal differences are the more important. IBM's Basic and BasicA, when loaded from disk, use code that's stored in ROM; the Corona GW Basic is entirely disk-based. Since the IBM ROM routines are not available on the Corona, you can't load IBM's BasicA.com and run it on the Corona. The IBM (Microsoft) Basic compiler makes some ROM calls as well, so programs written in Basic and then compiled with the IBM compiler may not run on the Corona.

Many compiled Basic programs do run, however. The popular "shareware" programs *PC-Talk III* and *PC-File*, for example, are supplied as compiled Basic programs, and both run as supplied. The IBM Basic compiler itself does not run correctly on the Corona.

The external differences between GW Basic and IBM's BasicA are relatively minor. GW Basic supports essentially the same command set as BasicA, including the graphics commands and the various sound-generating commands. There are, however, some minor differences in syntax, and Basic source programs written for the PC may require minor changes before they'll run on the Corona.

There were still some bugs in the GW Basic interpreter supplied with our test unit. One of these is important enough to note here: The *on key* commands, which enable a program to respond to interrupts from the function and arrow keys, do not work properly, and Basic programs that use this construct do not work as expected. A new release of GW Basic is supposed to correct this error.

Not a lot of serious application software for the PC is written in Basic, so the minor differences in Corona and IBM Basic are not likely to cause you much difficulty. You should, however, be aware of them.

Both CP/M-86 and the UCSD p-System run on the Corona, as does MS-DOS. Many other computers claiming IBM compatibility lack the ability to run the p-System. The large library of software written for this operating system includes such products as Context *MBA*, Applied Software Technology's *Versa-Form*, and State of the Art's accounting packages. You might find an inability to run this software a significant disadvantage of other machines.

The Corona runs three different versions of the UCSD p-System—Volition Systems's ver-

A personal library for your

*IBM PC is a registered trademark of International Business Machine Corporation.

Best Sellers

Inside the IBM PC: Access to Advanced Features and Programming by Peter Norton

The most widely-read author on the IBM PC gives the reader a look inside the workings of the computer and the advanced features it can provide. This best seller includes the fundamentals of the microprocessor, the operating system and PC DOS. In a very readable book, Peter Norton takes you beyond the machine, interprets its languages for you and introduces you to ROM.
1983/320pp/paper/ISBN 0-89303-556-4/$19.95
Book/Diskette Package ISBN 0-89303-561-0/$79.95

IBM PC: An Introduction to the Operating System, BASIC Programming and Applications, Revised and Enlarged by Larry Joel Goldstein and Martin Goldstein

An updated and expanded version of what has become the classic self-study book for the IBM PC. New chapters on BASIC emphasize the importance of structuring and planning programs and important new information on debugging. Games and graphics are included and the section on random access has been expanded. Includes DOS 2.0.
1983/448pp/paper/ISBN 0-89303-530-0/$17.95
Book/Diskette Package ISBN 0-89303-527-0/$39.95

IBM PC/XT Assembly Language: A Guide for Programmers by Leo J. Scanlon

An introduction to the fundamental principles of microprocessors (specifically the 8088), numbering systems, and assemblers. The book first outlines the steps necessary to create and run assembly programs and then describes the entire instruction set of the 8088 microprocessor.
1983/384pp/paper/ISBN 0-89303-241-7/$19.95
Book/Diskette Package ISBN 0-89303-535-1/$49.95

Personal Computer

MS DOS and PC DOS: User's Guide by Peter Norton

The authority, Peter Norton, gives an introduction and explanation of the Microsoft Disk Operating System precisely right for the beginner. It provides examples and instructions not available in any other book. Just off the presses, MS DOS and PC DOS will be the next Norton best-seller.

1983/250pp/paper/ISBN 0-89303-645-5/$15.95

Firsts

8087 Applications and Programming for the IBM PC and Other PCs by Richard Startz

New on the shelves and on the minds of 8087 microprocessor users, the pages begin with a non-technical introduction and turn into very detailed technical descriptions of the potential application. It includes sections on Advance Instruction Set, Non-linear methods, Statistical Analysis and Program Canning.

1983/250pp/paper/ISBN 0-89303-420-7/$19.95

Communications and Networking for the IBM PC by Larry E. Jordan and Bruce W. Churchill

This book brings together data communications applications and the IBM PC. It includes asynchronous and synchronous communications and a complete study of local area networking. A full chapter on communication codes and controls is included to demystify the language.

1983/225pp/paper/ISBN 0-89303-385-5/$16.95

It's Basic

Handbook of BASIC for the IBM PC by David Schneider

This book for the layman presupposes no knowledge of BASIC and clearly translates the BASIC reference manual supplied with the IBM PC into completely comprehensible terms. Organized by BASIC programming statements, it allows the user to go directly to the desired information without confusion or delay.

1983/350pp/paper/ISBN 0-89303-506-8/$19.95

IBM PC/XT Owner's Manual: A Practical Guide to Operations by Barbara Lee Chertok, Dov Rosenfeld, and James Stone

This entry level book provides easy instructions on the operations of the IBM PC. After reading this new guide with clear, concise examples, the PC user is up and running on the machine in a minimum amount of time — without programming classes.

1983/128pp/ISBN 0-89303-531-9/$12.95

Advanced BASIC and Beyond for the IBM PC by Larry Joel Goldstein

A complete guide to the advanced skills of BASIC programming, files, graphics, event-trapping, machine language and subrouting. A must for the IBM PC programmer.

1983/400pp/paper/ISBN 0-89303-324-3/$19.95
Book/Diskette Package ISBN 0-89303-325-1/$39.95

Business

Business Problem Solving with the IBM PC & XT by Leon Wortman

Business professionals will make use of the dozens of computer programs specifically designed for decision making and problem solving. Source codes are included in BASIC, with many also in Pascal.

1983/350pp/paper/ISBN 0-89303-282-4/$19.95

Business Applications for the IBM PC by Steven Zimmerman and Leo Conrad

First time users will find this guide to the IBM PC ideal. It offers step-by-step instructions on the use and customization of existing business software programs.

1983/350pp/paper/ISBN 0-89303-243-3/$17.95

Personal Interests

Games, Graphics, and Sound for the IBM PC by Dorothy Strickland, Dennis Rockwell, and Kevin Bowyer

This guide teaches how to program in BASIC, Pascal and FORTRAN to create graphics and sound for the IBM PC. It illustrates in 70 working examples how to integrate sound and graphics into animation. And, it includes an example of how to create a simple video game.

1983/288pp/paper/ISBN 0-89303-469X/$18.95
Book/Diskette Package ISBN 0-89303-470-3/$64.95

BASIC Engineering and Scientific Programs for the IBM PC by Philip Wolfe and C. Patrick Koelling

A source of BASIC programs written exclusively for on-the-job use by engineers and scientists to provide important computer techniques for problem solving and data manipulation.

1983/320pp/paper/ISBN 0-89303-330-8/$19.95
Book/Diskette Package ISBN 0-89303-331-6/$39.95

Pascal for the IBM PC: IBM DOS Pascal and UCSD p-System Pascal by Kevin Bowyer and Sherryl Tomboulian

This is the first written word on combining the IBM PC and Pascal Programming. Soon it will be the last word for anyone who wants to break the limits of BASIC programming. An emphasis on graphics and sound applications.

1983/320pp/paper/ISBN 0-89303-280-8/$17.95

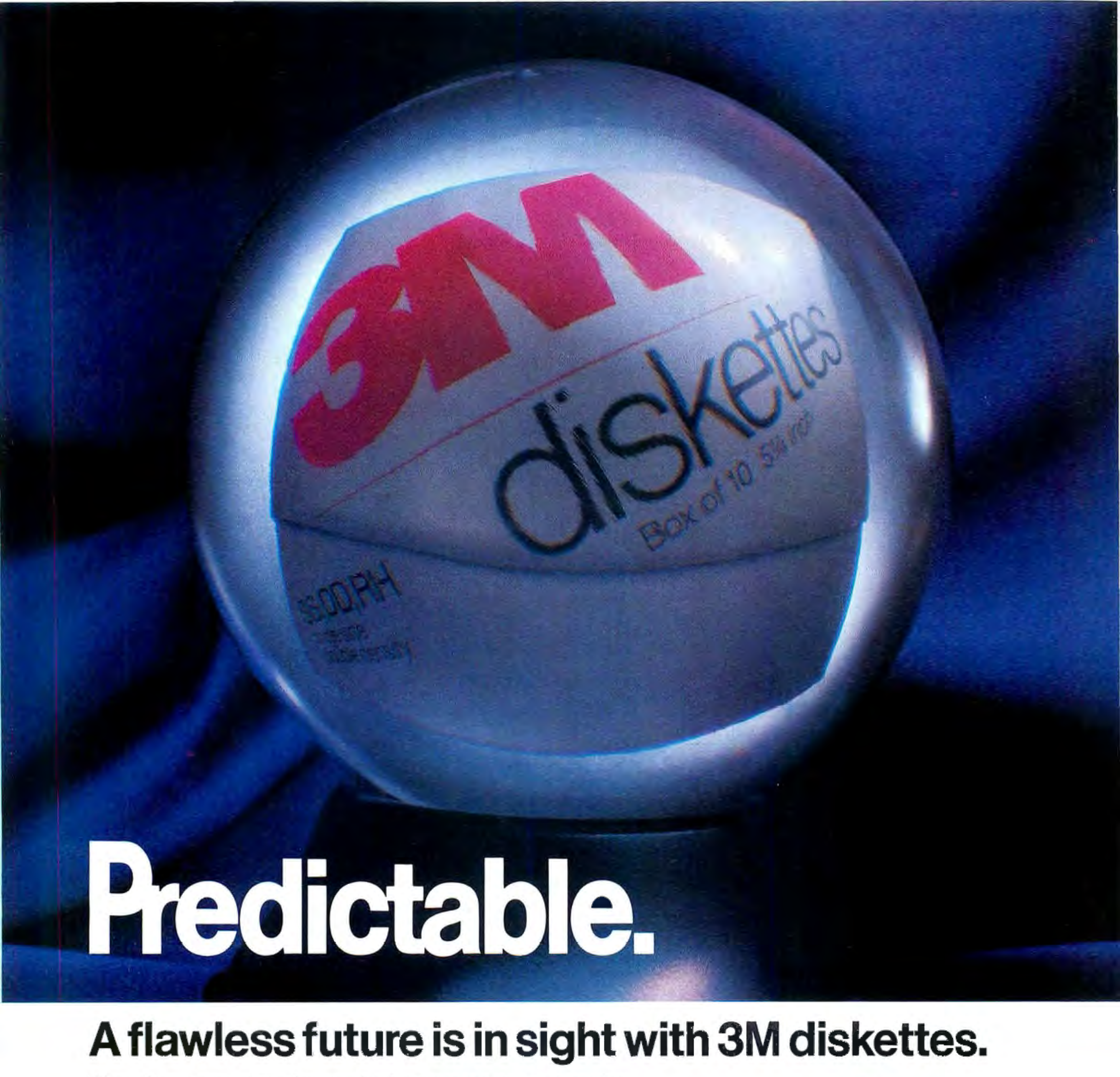
BRADY

Your publisher of books for the IBM PC

Brady IBM PC Bookware™ is available at Computerlands, IBM Product Centers, Sears Computer Centers, Entre Computer Centers, B. Daltons, Waldens, Crown Books and other fine bookstores and computer retailers.

If you are unable to find the book of your choice call toll-free 800 638-0220 for information.

Robert J. Brady Co. A Prentice-Hall Company Bowie, MD 20715



A flawless future is in sight with 3M diskettes.

When it comes to keeping track of precious data, predictable means reliable. Being able to count on every diskette, every time. At 3M, reliability is built into every diskette. We've been in the computer media business for over 30 years. And we've never settled in. We're constantly improving and perfecting our product line, from computer tape and data cartridges to floppy disks.

3M diskettes are made at 3M. That way, we have complete control over the entire manufacturing process. And you can have complete confidence in the reliability of every 3M diskette you buy.

Look in the Yellow Pages under Computer Supplies and Parts for the 3M distributor nearest you. In Canada, write 3M Canada, Inc., London, Ontario. If it's worth remembering, it's worth 3M diskettes.

3M hears you...

3M

sion 2.0, IBM's (Softech's) version 4.0, and Network Consulting's version 4.1. All three function perfectly with the Corona hardware and use its extra memory for RAM disks and its interrupts to maintain a print spool.

The Machine Itself. It appears that the Corona is compatible enough with the PC to satisfy the needs of most potential users. The rest of this evaluation focuses on features, performance, and cost as compared with similar IBM systems.

The desktop unit is about an inch deeper than the PC but otherwise looks just like it, with a front panel that provides space for two disk drives side by side on the right and a logo on the left. The unit is solidly built and attractive.

The portable is in the sewing-machine class; that is, it's really a transportable. It weighs nearly thirty pounds and is just small enough to fit under an airline seat. It too is pleasing to look at, with its display on the left and its drives mounted vertically on the right.

The cover is attached to the frame by four Phillips head screws. With the screws removed, the cover lifts off easily to reveal the computer's inner workings. The four expansion slots are located on the left side of the cabinet, as are the RAM sockets, which sit below any expansion boards that may be placed there.

Expansion boards in the Corona are mounted running from front to back and are provided with vertical support on the back plane. The Corona accepts expansion cards up to fourteen inches in length; thus there's room enough for any card that can be used in the IBM.

Corona's RAM is organized in banks of 128K, each requiring sixteen 64K RAM chips; the PC organizes its memory in banks of 128K, and, because its memory is parity-checked, requires nine 64K chips for each of these banks.

A bank of switches used to register the current hardware configuration—the amount of memory, the type of display, the number of drives, and hardware floating point—is located in an area cut out behind the disk drives, just in front of the system's quiet fan. A light-pen connector is located in the same cutout, along with a pair of jumper blocks used to configure the serial port as DTE or DCE.

The motherboard contains a socket for an 8087 coprocessor, but, in contrast to the PC—where the 8087 socket is easily accessible once the cover is removed—the Corona's 8087 socket is hidden beneath one of the disk drives, and further disassembly of the unit is required to get at it.

A large cage in the right rear corner of the system unit houses the Corona's 110-watt power supply. The standard PC has a 63-watt power supply, while the XT has 130 watts. The floppy-disk controller is built into the motherboard, and connectors for two floppies protrude from the center of the unit. With the hard disk in place, one of these connectors remains unused, but there is no convenient way to bring it outside the case to mount another floppy disk drive.

Corona's built-in display adapter, output ports, and disk controller provide functions that require the support of peripheral cards on the PC. The three slots that the PC normally uses for these functions are free in the Corona. Furthermore, the Corona's 512K on-board RAM capacity is double that of the IBM; PC users would have to use a RAM expansion card or combination card to get the same amount of memory. Thus the Corona with four slots is somewhat more expandable than the PC with five. (The hard disk requires that its own separate controller be mounted in one of the slots.)

The Corona forgoes the diagnostic self-testing of the PC. As a result, it boots up almost immediately, even when loaded with RAM. Despite the lack of power-on self-testing and parity-checking, the Corona we tested functioned reliably and predictably.

Although the Corona does not do diagnostic tests each time you turn it on, Corona's engineers have not ignored these tests. A memory test program is included with Corona's version of MS-DOS. It displays a diagram of the populated and unpopulated RAM sockets and shows you which ones are okay.

The ten-megabyte hard disk is easily and quickly formatted by Corona's software, and performs beautifully. Corona sells the same unit separately as a second-source drive for the PC. The unit tested had an occasional whine, apparently because of a bad bearing, but its performance never faltered. The floppy disk drive made about as much noise as one would expect from IBM drives.

The Corona's monochrome display is outstanding. Its characters, formed in a sixteen-by-thirteen matrix, are clear and precise. The character set is identical to the PC's, and all characters can be displayed with any combination of the underline, blinking, reverse video, and high-intensity attributes.

As mentioned, the monochrome display adapter also supports high-resolution graphics output. The graphics have an extremely high 640-by-325 pixel resolution, and text can be mixed with graphics.

At the moment, unfortunately, your only access to Corona's built-in graphics is through the GW Basic interpreter. The graphics do not conform to IBM's standard, so no IBM products use them, and Corona provides no software support or documentation beyond GW

Basic. Several Basic demos supplied with the system take advantage of the graphics display, and it's truly first-class. It's unfortunate that no better support is available.

The monitor supplied with the desktop unit is an Amdek 310 bearing Corona's own label. This twelve-inch diagonal, green-screen unit, with brightness and contrast controls on its front panel, renders sharp text and graphic output. The portable units have a nine-inch screen with a similarly precise display.

The Corona keyboard's six-foot coiled cable attaches (and detaches) directly under the front panel. This arrangement makes the Corona keyboard more mobile than the IBM keyboard, which attaches at the rear. The Corona keyboard is manufactured by Key Tronic (the same keyboard is also available from Key Tronic as an alternative to the original PC keyboard). The caps-lock and num-lock keys have lights that indicate whether these shifts are in effect—a distinct improvement over the PC keyboard. The keys have a somewhat spongier feel than those on the IBM and lack its loud clicking, but the unit is fast and responsive.

Although the layout on our test unit was identical to the layout on the IBM keyboard, more recent models interchange the backslash key with the left shift and place the enter key directly to the right of the quotes key.

The hardware was not entirely bug free, but the one bug we found was rather minor. The on-board switch setting that was supposed to identify the system as having a color/graphics card with an eighty-column display did not work properly; when the machine was turned on with this switch setting, it came up with a forty-column text display. This was apparently the result of an error in the Corona ROM; the company says that it's been fixed.

Software and Documentation. A review of the Corona PCs would not be complete without an appraisal of the software included with them. The version of MS-DOS supplied with the computers includes all the standard MS-DOS utilities and has some additional features as well. The most important of these is support for a RAM disk. The operating system requires you to reserve at least 64K as ordinary memory, but, using the Config utility, you can designate as much additional RAM as you wish as a disk emulator.

The ten megabytes of Corona's hard disk can be distributed among as many as four separate partitions. You can specify how much memory you want assigned to each partition and the drive designators (A, B, or whatever) for each. Drive A can be either your floppy or one of your hard disk partitions—your choice. Additional MS-DOS utilities perform disk diagnostics and provide for backup and backdown of hard-disk files.

MS-DOS 2.0 was not available for Corona's hard disk at the time of this review, and

IBM's PC-DOS 2.0 does not recognize the Corona disk as a fixed disk. Corona says that MS-DOS 2.0 will be available to run on its hard disk by the time you read this.

Most of the software we tested that's designed to run in PC-DOS 1.1 works perfectly with Corona's hard disk—by treating the partitions of the disk as if they were ordinary (large) floppies. The sole exception is *TK!Solver*, which recognizes neither Corona's hard disk nor its RAM disk.

At the moment, Corona supplies no software for using the hard disk with any operating system but MS-DOS. NCI, which markets version 4.1 of the UCSD p-System, is planning to release software that will allow part of the hard disk to be used with the p-System, and support for a CP/M-86 partition may also be forthcoming.

The *PC-Tutor* package included with the Corona provides a gentle introduction to the use of the Corona and MS-DOS. It's thorough (it even covers the use of Exe2Bin) and may even be of value to experienced users. For example, its lesson concerning MS-DOS's template for editing command lines should make life considerably easier for users who are unaware of this facility.

Softword's *MultiMate* word processor makes the PC usable as a word processing machine by the vast army of operators familiar with the Wang word processor. This fast, powerful program has a built-in mail-merge capability and a useful document-management feature. However, not everyone will be pleased with its page-oriented design.

The documentation for the Corona system and its associated software comes in three separate plastic three-ring binders. One contains the documentation for MS-DOS and a guide to the hardware, the second is a GW Basic manual, and the third is a manual for *MultiMate*.

Price Comparisons. The recommended retail price of the Corona desktop unit in its minimal single-drive configuration is $2,595. The two-floppy unit costs $2,995, and the unit with the hard disk lists at $4,495. The portables are $50 less than equally equipped desktop units. At their recommended retail prices, the Corona units cost 15 to 20 percent less than comparable IBMs. But the Coronas include bundled software, and when its value is factored into the comparison, the Coronas can be considered 20 to 25 percent less expensive than comparable PCs.

The Company. In the wake of Osborne's recent declaration of bankruptcy, many buyers are understandably concerned about the stability of the companies from which they buy their equipment. Although the Corona PCs represent the first venture of Corona Data Systems into computer production, the company is well established as a manufacturer of hard-disk subsystems for the PC, the Apple, and other computers. The company was founded by Dr. Robert Harp, the cofounder of Vector Graphic.

Corona's computers are sold by dealers, and supplied through a network of full-service distributors. Coronas can be serviced by local dealers and distributors or any Xerox Computer Center.

Summary. The desktop Corona withstood prolonged use without complaint. The combination of hard disk and floppy was ideal for most applications, but a second floppy drive would have been useful for running some copy-protected software. With two floppy drives and MS-DOS 2.0, the Corona would be an ideal machine. Unfortunately, the hard-disk unit is not currently available with half-height floppies, and MS-DOS 2.0 is not yet here.

Since IBM offers no portable, Corona's portables will be particularly attractive to buyers who'd like a more transportable machine. The prices for the Corona portables are highly competitive with those of several other portable IBM-compatible computers currently on the market. And the machines are more expandable and PC compatible than most. ▲

HEROISM IN THE MODERN AGE

THE ROLE-PLAYING GAME OF TODAY

© *Copyright 1983* **Pacific Infotech Corp.**

- The first truly complete computerized role-playing system.
- A multi-character role-playing game which is vast in scope, allowing hundreds of incredibly detailed characters of your own creation.
- A sophisticated simulation of modern life and human psychology.
- A test of creative thinking and imagination.
- A game designed especially for the IBM PC*
- The only system realistic enough to be believable in *real-life* rather than a fantasy setting.

Your Starter Pack contains 2 full diskettes and detailed manuals. The game has 2 parts:

1. BASIC MODULE — which allows you to create characters and guide them through their lives. You can have the computer create characters — giving them names, a family background, and psychological and physical makeup — or you can custom-design characters, perhaps to represent people in real-life.

2. PROJECT CONTACT
Scenario #1 — which allows you to take characters from the Basic Module or pre-created characters on a secret mission to save the United States from a mad scientist and a terrorist plot. You must use all of your character's abilities and skills and technological tools to find clues, persuade allies, question suspects, fly helicopters and fight battles. You have only 60 hours to complete your mission...

HEROISM IN THE MODERN AGE is expandable, with other modules available.

MORE THAN JUST A GAME...
PROJECT CONTACT Scenario #1

at your computer store

SYSTEM REQUIREMENTS:
IBM PC with one double-sided or two single-sided disk drives, an 80-column monitor, and 64K RAM (96K for DOS 2.0).*
Color graphics/adaptor optional
IBM is a trademark of International Business Machines Corp.

Pacific Infotech Corp.

10850 Wilshire Blvd.,
Suite 800
Los Angeles, CA 90024
U.S.A.


NOW YOUR COMPUTER FITS THE ORIGINAL ARCADE HITS.
TI 99/4A
IBM PC
COMMODORE 64
COMMODORE VIC 20
DIG DUG
FROM ATARISOFT
ROBOTRON: 2084
FROM ATARISOFT
STARGAT
FROM ATARISOFT
PAC-M
FROM ATARISO
CENTIPEDE
FROM ATARISOFT
DONKEY KONG
BY NINTENDO*
ATARISOFT

Now the excitement of original arcade graphics and sound effects comes home to your computer.

Introducing ATARISOFT.™ A new source for computer software.

If you own a Commodore VIC 20 or 64, a Texas Instruments 99/4A, an IBM or an Apple II, you can play the original arcade hits.

DONKEY KONG by Nintendo, CENTIPEDE,™ PAC-MAN, DEFENDER, ROBOTRON: 2084, STARGATE and DIG DUG. (On the TI 99/4A you can also play Protector II, Shamus, Picnic Paranoia and Super Storm.)

So, start playing the original hits on your computer.

Only from ATARISOFT.

Some games also available on ColecoVision and Intellivision.

ATARISOFT™

Now your computer fits the arcade hits.

DONKEY KONG, Mario and NINTENDO are trademarks and © Nintendo 1981, 1983. PAC-MAN and characters are trademarks of Bally Midway Mfg. Co. sublicensed to Atari, Inc. by Namco-America, Inc. DEFENDER is a trademark of Williams Electronics, Inc., manufactured under license from Williams Electronics, Inc. ROBOTRON: 2084 is a trademark and © of Williams 1982, manufactured under license from Williams Electronics, Inc. STARGATE is a trademark and © of Williams 1981, manufactured under license from Williams Electronics, Inc. DIG DUG is created and designed by Namco Ltd. manufactured under license by Atari, Inc. Trademarks and © Namco 1982. PROTECTOR II is a trademark of Synapse Software Corporation, manufactured under license by Atari, Inc. SHAMUS is a trademark of Synapse Software Corporation, manufactured under license by Atari, Inc. PICNIC PARANOIA is a trademark of Synapse Software Corporation, manufactured by Atari, Inc. SUPER STORM is engineered and designed by Synapse Software Corporation, manufactured under license by Atari, Inc. ATARISOFT™ products are manufactured by Atari, Inc. for use on the above referenced machines and are not made, licensed or approved by the manufacturers of these machines. COMMODORE 64, VIC 20, TEXAS INSTRUMENTS 99/4A, IBM, APPLE, COLECOVISION and INTELLIVISION are respectively trademarks of Commodore Electronics Limited, Texas Instruments, International Business Machines Corp., Apple Computer, Inc., Coleco Industries, Inc. and Mattel, Inc. A Warner Communications Company.

Complete this coupon and we'll keep you up to date on the newest hits from ATARISOFT.™

Name

Address

City State Zip

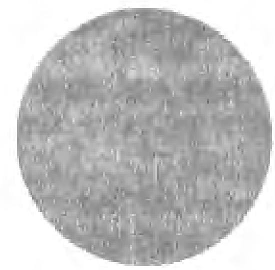
Telephone

PRODUCT OWNED: (Check one)

1 ☐ TI-99/4A
5 ☐ Commodore Vic 20
2 ☐ IBM PC
6 ☐ Intellivision
3 ☐ Commodore 64
7 ☐ Apple II
4 ☐ ColecoVision
8 ☐ ________

Mail to:
Atari, Inc., P.O. Box 2943,
So. San Francisco, CA 94080.

ASM 19



THE C SPOT

A TUTORIAL ON THE C PROGRAMMING LANGUAGE

by Rex Jaeschke

Welcome to the wonderful world of C. During the coming months we will look at the statements and constructs of the C programming language. Each new construct will be introduced with an example and a corresponding explanation. Where possible, the examples will be complete programs. The reader is encouraged to enter, run, and modify these examples. The only way to master any language is to write programs in it, run them, and debug them.

All program examples have been taken directly from the manuscript text file and tested using the DeSmet C Development System v2.2 (C Ware, 970 West El Camino Real, Sunnyvale, CA 94087; 408-735-7507) running on a PC with PC-DOS 1.1 (and 2.0 where applicable).

C is a midlevel, or systems, language. That is, C fits somewhere between assembler and high-level languages. On the one hand it has many capabilities traditionally found only in assembly languages. It allows the programmer to get close to the host-machine architecture and instruction set by using constructs such as ++ and −− for auto increment and decrement. Conditional compilation directives are supported, as are bit-field operators and address arithmetic. On the other hand, C supports high-level language constructs such as if-else, while, for, and case statements.

C was developed in 1972 at Bell Labs by Dennis Ritchie. The Unix operating system, another Bell Labs development, was then rewritten in C. Unix and C have been closely associated ever since. Every Unix and Unix-like operating system package includes a C compiler.

C is not a language for the beginner. It was designed by and for professional programmers. You'll need some advanced programming

Rex Jaeschke is a consultant, educator, and freelance writer in the Washington, D.C., area. His interests include languages, communications, and computer-aided education.

experience before you'll be able to make full use of some of C's more powerful capabilities—particularly pointers and structures.

A major reason that software developers use C is the ease with which properly designed programs can be ported across different hardware and system-software environments. With rapid development of newer and faster processors, software houses must design programs that will outlive the current systems. The time and cost of rewriting software for new target machines is prohibitive if the vendor is to retain a reasonable market share. Digital Research, Microsoft, and VisiCorp are among the many software developers using C as a tool to enhance program portability.

Designing portable programs takes care and discipline. Using C, in itself, does not guarantee portability. It is easy to write machine-dependent C code. The decision to make a program portable must be made before the program is designed. Many different aspects of portability must be reviewed. A future installment will cover portability considerations in detail.

C can provide power similar to that of assembly language and maintain machine independence at the same time because it has no provision for data input and output. These capabilities are provided by compiler-vendor or user-written routines that are invoked from C programs. This may seem awkward at first, but once you have written a few programs it seems a simple extension to the structured-programming approach.

The definitive reference for C is *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie (Prentice-Hall, 1978). Throughout this series, they and their book will be referred to as K&R. K&R has a tutorial format and encourages the reader to program examples and problems. It assumes prior programming experience and is more of a user's guide than a reference manual. Most C compiler manuals refer to K&R for C language specifics. Any reader seriously interested in evaluating or using C will want a copy of this book. Fortunately, the authors have defined a series of commonly used I/O routines whose naming and calling conventions have been accepted as a standard by most compiler vendors.

C is a simple language with only about thirty keywords. Because of the power of these basic statements and constructs and the ability of programmers to extend the language by using callable functions while maintaining portability, there is little need for language extensions. However, they do exist in some compilers and vary from one vendor to another. Programmers should be aware of any nonstandard conventions in production compilers they use.

Programmers new to C may initially find programs hard to read because almost all code is written in lowercase. Lower and uppercase letters are treated by the compiler as different character sets. In fact, keywords must be written in lowercase. If you have ever wondered what all those special punctuation marks on an ASCII keyboard are for, you'll be interested to learn that one of their uses is in writing C programs. C uses almost all of them for one reason or another. Keywords are reserved, and the language has been designed to be unambiguous, which allows C compilers to be easily written. For example, the assignment operator is =, while the equality operator is ==.

C encourages, indeed forces, a structured, modular approach to programming with callable subroutines, called functions. Functions can be compiled separately, with global references and other resolutions made at link time.

Unlike Pascal, C is weakly typed. It permits the programmer to do seemingly silly things, particularly in mixed-mode arithmetic. This allows both powerful tricks and nasty errors. C code can be tight and obscure to the point of being unreadable and, therefore, unmaintainable. However, with care and discipline, a programmer can generate clearly formatted and easy-to-read code.

If C is so great, why doesn't everybody use it? Well, like other languages, C is not all things to all people. For many applications, Cobol, Fortran, and Basic do fine. One aspect of C that discourages commercial programmers is the lack of support for group moves or compares of character strings, arrays, and structures. Such capabilities must be provided by external functions, which generally are provided in the compiler vendor I/O library. Unless a programmer really needs the special capabilities provided by C, he or she is unlikely to use it in production.

Program Structure. Let's begin by looking at the basic structure of a C program.

```
/* -- smallest.c     The smallest possible C program -- */
main ()
{
}
```

A C program consists of one or more functions that may occur in any order in a source code file. A program must contain a function called *main**. *main* is a special function name that indicates where the program is to begin execution. There is no differentiation between the main program and any subprograms other than their names; they all have the same structure and are referred to as functions.

The parentheses following the function name surround the dummy argument list. The parentheses are required, even if no arguments are expected. The function *main* can have arguments but in this case does not. Passing arguments to *main* from the operating system command level will be discussed in detail later.

Comments are delimited by /*, which may occur anywhere a blank or newline can. We'll learn more about the newline character later. Braces are required for each function, and any executable code for that function must appear inside them. Statements are executed in sequential order unless control or looping statements dictate otherwise. A program terminates when it reaches the closing brace of the function *main*.

C is a free-format language. Spaces, tabs, and blank lines can and should be used liberally to improve readability. They are ignored by the compiler unless they are part of a quoted literal string. A compact form of the previous program is

```
main() {}
```

Although both of the programs will compile without error, they have no executable code and therefore do nothing.

```
/* --- small.c Smallest C program with 2 functions ---*/
main ()
{
	sub ();
}
/* -- sub has no executable code -- */
sub ()
{
}
```

C encourages programmers to break large programs into a number of smaller functions. A source file may contain one or more functions, which can be defined in any order. Function definitions cannot be nested. That is, the function *sub* must be defined outside of the braces delimiting the function *main*. A function is invoked by naming it, along with a list of arguments surrounded by parentheses. Arguments are optional, but the delimiting parentheses are not.

Invoking a function is considered a statement. Each C statement must be terminated by a semicolon. Pascal programmers should note that the semicolon is a statement terminator, *not* a statement separator as it is in Pascal. When function *sub* terminates at its closing brace, control is passed to the statement immediately following the one that invoked *sub*.

Function and variable names traditionally are written using lowercase letters. Notice how spaces, tabs, and blank lines are used to make the program more pleasing to the eye. Also, the opening and closing braces are aligned to indicate better each function's scope. K&R go to some pains to recommend a program formatting style. Although their recommended style is widely used and works well, the programmer is encouraged to experiment. Whatever style you adopt, be consistent.

```
/* -- welcome.c produces output from C -- */
main()
{
	printf ("Welcome to the world of C.\n");
}
```

Formatted Screen Output. The C language has no input or output statements. Such capabilities are provided as part of a run-time library of functions that are supplied by the compiler vendor or written by the user. *printf* is such a library function and prints formatted output to the standard output device, which is typically the user's terminal. PC-DOS 2.0 allows standard output to be redirected to a device or file other than the console.

The character string to be printed is enclosed in double quotes. The \n, which is a special *single* character, is C notation for newline. *printf* never prints a newline automatically, so it may be invoked multiple times to print an output line a piece at a time. The newline is a line-feed character and typically is converted to a CR/LF pair on output. The backslash is an escape-character prefix that indicates the following character is to be treated specially. Other common sequences are \b for backspace, \t for tab, \" for double quotes, \f for form feed, and \\ for the backslash itself. This mechanism allows character-set independence for these special characters. At least one C compiler correctly interprets uppercase escape-sequence letters such as \N and \T. There is no guarantee that this is true of other compilers.

The previous program could be written as

```
main ()
{
	printf ("Welcome ");
	printf ("to the world of C.");
	printf ("\n");
}
```

or as

```
main ()
{
	printf ("Welcome "); printf ("to the world of C.");
printf ("\n");
}
```

All three examples are equivalent; however, the first one is more readable and will be executed faster, since it involves only one call to *printf*. The third example shows that multiple statements may exist on the same source-code line, provided each is terminated by a semicolon. This practice is discouraged because it makes the code less readable.

When a function is invoked, all its arguments must be specified on the same source-code line. The maximum length of a source-code line and a quoted literal string is compiler-dependent.

```
/* -- variable.c define and print an integer variable -- */
main ()
{
	int year;
	year = 1983;
	printf ("This year is %d.\n",year);
}
```

Variable names may be of any length, but C recognizes only the first eight characters. Some compilers recognize longer names. Avoid this language extension if portability is important. Names are made of letters and digits, and the first character must be a letter. The underscore (_) is allowed and is considered a letter. Again, upper and lowercase characters are different. For example, *year* and *YEAR* are two different variables. Traditionally, variable names are written in lowercase.

Variables must be declared explicitly. A declaration consists of a type followed by a list of variables that have that type, such as

```
int start,end,inc;
```

year is declared to be of type int, which means integer. The size of the number that can be stored in an int variable is machine-dependent, as are most C data types.

Other data types include char (character), short (short integer), long (long integer), unsigned integer, float (single-precision floating point), and double (double-precision floating point). The amount of storage space allocated to variables of each data type may vary considerably with dissimilar machines. On sixteen-bit machines, int and short are generally sixteen bits, while on thirty-two-bit systems, int and long are usually thirty-two bits. In fact, it is possible that int, short, and long variables are all allocated the same amount of storage. Usually, a float variable uses four bytes and a double variable uses eight, although float and double may have the same precision. Each character variable may use six, seven, or more bits. On eight-, sixteen-, and thirty-two-bit machines, usually eight bits are allocated.

Variables are assigned values with the symbol "=". The value assigned may be a constant, the value of another variable, the result of a function, or an expression involving any or all of these.

The *printf* function call has two arguments this time, a character string and an integer variable *year*. The string serves as a format or edit mask and contains a special sequence, %d, which causes the first argument after the string to be interpreted and printed as a decimal integer. Other masking sequences exist for each data type, and the masks present in the string determine the number and type of arguments that are expected after the first argument. The value of *year* will be inserted into the text string when it is printed, followed by a newline, indicated by the \n. The output generated is

```
This year is 1983.
```

Note that *year* is defined inside the braces of *main*, an area reserved for executable code. The definition int *year;* can be considered executable, as storage for the variable year is allocated at run time and *not* when compiled. Variables declared within a function are local to that function and are not accessible to other functions. They are created *each* time their parent function is invoked and disappear when that function is exited. For this reason, they are known as automatic variables. Automatic variables should be initialized explicitly, or else they will contain garbage.

Storage for automatic variables is allocated on the hardware stack. Therefore, too many concurrent automatic variables, particularly arrays and structures, will cause stack overflow. The size of the stack may be determined by the compiler and/or linker used. Let's look at a set of variable declarations and the 8086/8088 code they generate.

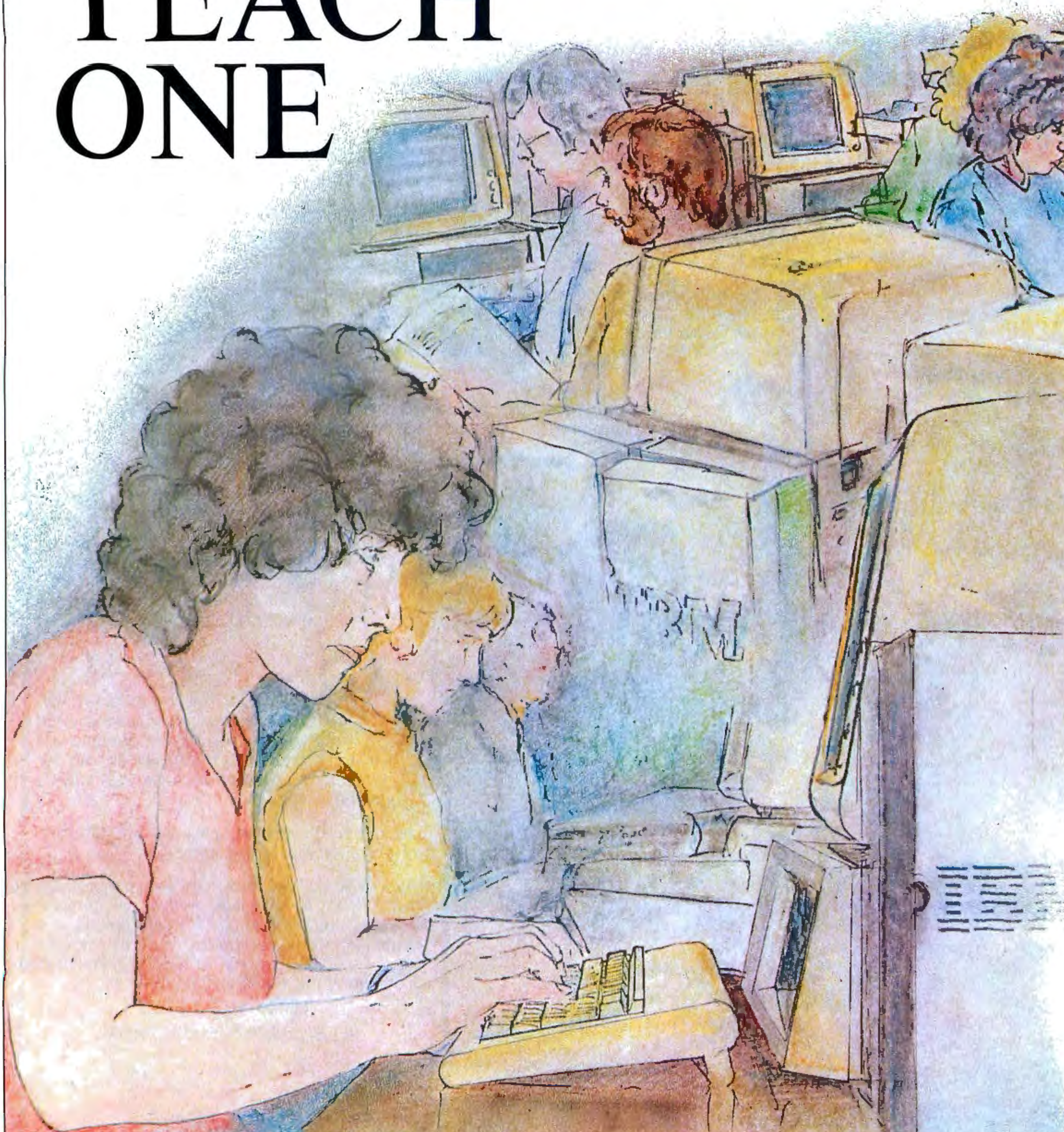
```
sub ()
{
                            /* PUSH BP */
                            /* MOV BP,SP */
        char c1,c2;
        short s;
        int i;
        long l;
        unsigned u;
        float f;
        double d;
                            /* SUB SP,24 */
}
                            /* MOV SP,BP */
                            /* POP BP */
                            /* RET */
```

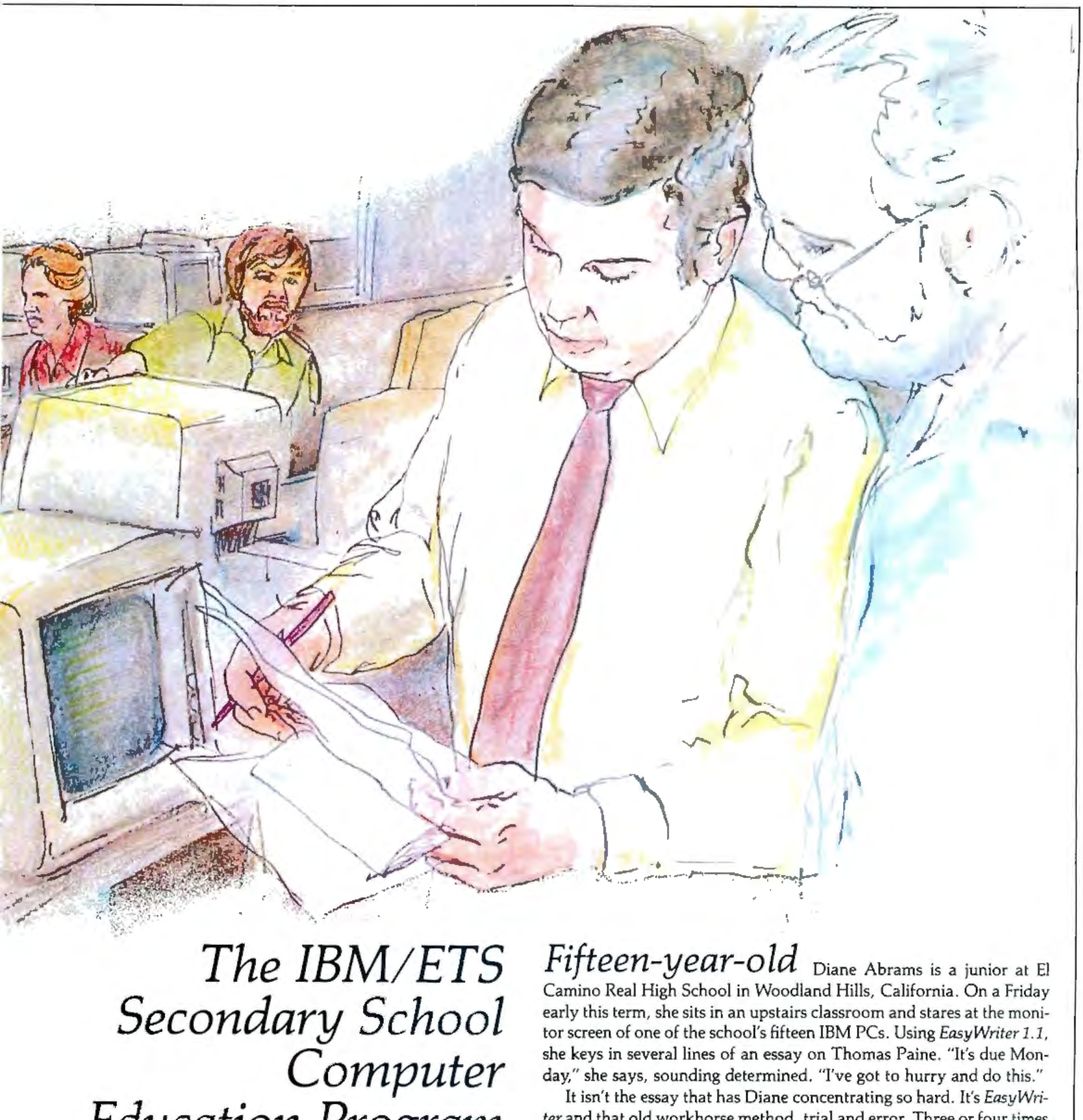
Twenty-four bytes are reserved on the stack for the eight automatic variables. One for each char; two each for short, int, and unsigned; four each for long and float; and eight for double. At the end of the function, this storage space is released by the adjustment of the stack pointer to its value on input.

One interesting aspect of the DeSmet compiler is that if the declaration for the char variable c2 is omitted, the compiler still reserves twenty-four bytes. That is, it forces word (even byte) alignment on the stack, at the possible expense of a wasted byte. This has little effect on 8088-based systems because they use byte operations. However, many 8086-based programs could benefit from significant speed improvements because they typically do a lot of automatic variable (and hence stack) manipulation.

Let's hope this introduction has whetted your appetite for more information about C. Next time we'll look at some of the looping and branching constructs and introduce symbolic constants and character I/O. Future installments will be more "meaty," particularly in reference to code-generation efficiency, and will include machine-code examples as appropriate. ▲

EACH ONE TEACH ONE





The IBM/ETS Secondary School Computer Education Program

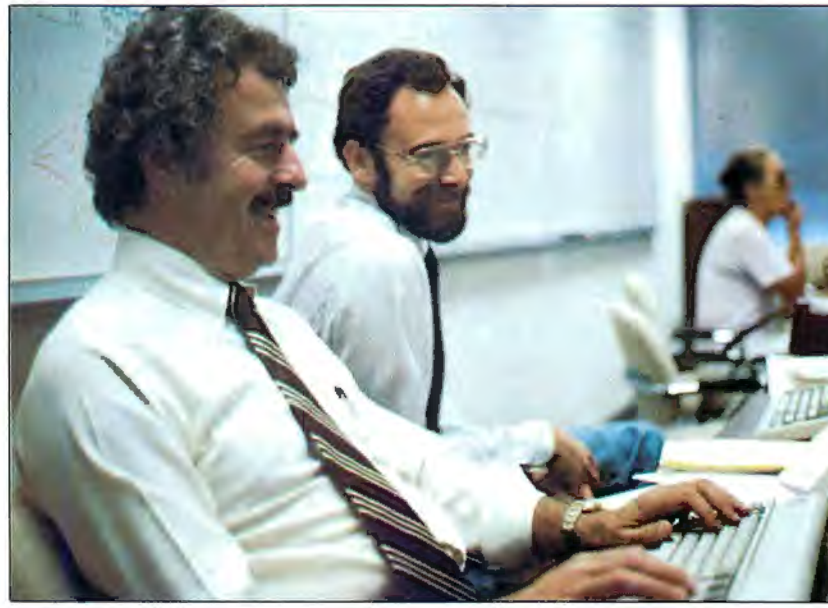
by JoAnn Levy

Illustration by Barry Hoff

Fifteen-year-old Diane Abrams is a junior at El Camino Real High School in Woodland Hills, California. On a Friday early this term, she sits in an upstairs classroom and stares at the monitor screen of one of the school's fifteen IBM PCs. Using *EasyWriter 1.1*, she keys in several lines of an essay on Thomas Paine. "It's due Monday," she says, sounding determined. "I've got to hurry and do this."

It isn't the essay that has Diane concentrating so hard. It's *EasyWriter* and that old workhorse method, trial and error. Three or four times she forgets she's in insert mode and accidentally loses a couple of lines, but she's learning. This is only Diane's second experience with the computer and software. Yesterday she focused on the documentation, but now she's impatient to conquer the program itself. "In the end, doing the essay this way will save me a lot of time," she says. "I won't have to type it."

A series of decisions and events in the spring and summer of 1982 are what's making it possible for Diane Abrams to cut short the drudgery of typing her English assignment. One of thousands of high school students in three states who are being affected now by what happened then, Diane is probably blissfully unaware of the series of events set in



(Above) The last day of classes at Pepperdine, then the PCs will go back into the waiting boxes and be on their way. (Above right) Jack McManus, left, and Brian Stecher, right, from the local ETS office, going over the teaching curriculum. (Far right) Terry Cannings, one-half of the McManus & Cannings team. (Right) Tony Cline, program director for ETS: "I'm impressed and pleased to be associated with such a quality operation."

motion more than a year ago. All she knows—all she needs to know—is that she has a PC in front of her and that she is rapidly becoming skilled in its use.

How the PC got there is the real story, a story with lots of action and a huge cast.

The action begins many months ago at IBM's corporate headquarters in Armonk, New York.

Fall 1982. Inspired by the so-called Apple bill (California's law AB 3194), which allows computer manufacturers to donate micros to schools in exchange for tax credit, IBM's board of directors decides it wants to make a planned contribution to precollege education.

"The result we are aiming at," IBM vice president and chief scientist Lewis M. Branscomb will say later, "is a nation better prepared for the technological opportunities in the years ahead."

Most of IBM's previous philanthropic programs have been directed toward postsecondary education and have been handled, appropriately enough, through the company's office of university relations. Confident that the existing group will work effectively in this situation as well, the board of directors passes the charge of implementing its decision to the university relations people. This group, in turn, calls in the Educational Testing Service (ETS) of Princeton, New Jersey, with whom they've collaborated successfully in the past.

The buck stops there.

ETS is a nonprofit corporation developed following World War II. It was at this time that a number of institutions became interested in admissions testing and in developing a research base to enhance testing activities, which were seen as an important factor in decision making. ETS accepts IBM's mandate, agreeing to advise the corporation about the most effective ways to put its contribution to use.

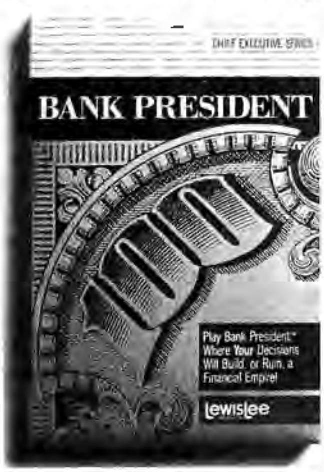
"We had been watching the Apple bill with some interest," ETS's Tony Cline, director of the IBM program, would remember, "and we were pleased that substantial amounts of hardware would be made available to schools. But we were concerned that the sprinkling of one machine per school might cause problems. Such distribution can raise expectations and promote terrible disappointment."

ETS has for some time now been holding seminars on computer use in elementary and secondary schools. This means that the organization has access to a network of seminar graduates who are committed to and knowledgeable about the use of microcomputers in the schools. Drawing on its own experience and theirs, ETS has come up with a working notion of what components constitute a successful computers-in-education program.

Over a period of months, ETS develops its plan for a successful program for IBM, negotiating the major points as they emerge. It will be important, ETS decides, to provide hardware, software, and teacher training. It will also be vital to create a network of schools, with a coordinator who can offer support in evaluating new software and in-

Play BANK PRESIDENT™, where your decisions will build-or ruin-a Financial Empire!

Compete with your friends or play by yourself in this easy-to-use, hard-to-master game—designed by management consultants, bankers, and software experts to give you the management challenge of your life!



Bank President—first in the Lewis Lee Chief Executive™ Series of exciting management action games.

Do you feel you have what it takes to run a large organization? Well, here's your chance! See how well you can do under the pressures and rewards top executives face. Lewis Lee's new Chief Executive™ Series—starting with Bank President—puts YOU in charge.

Executive Competition

As the Bank President, you choose and name the kind of outfit you want to run—a centralized business-oriented bank, or a retail-oriented one with many branches. You make critical business decisions:

- Choose your product lines
- Raise or lower employee salaries

Bank President lets you monitor your progress with clear graphic output.

- Issue and redeem stocks and bonds
- Manage your investment portfolio
- Speculate on interest rates or hedge against fluctuations
- Borrow from the Federal Reserve
- Set loan and deposit interest rates

IBM and Apple Versions

Bank President is fun to play by yourself or against your friends. You can even form teams, with each team managing its own bank and competing against the other teams. The winner is the one whose bank creates the most wealth for its stockholders during the course of the game.

Bank President's menu-driven design lets you focus on management action, not computer operation. Easy-to-read graphs, pie charts and bar charts track your success and that of competitors. The user's manual even gives you tips on how to outperform opponents. And Bank President stays challenging—it changes every time you play.

Excitement through Realism

You learn as much about opponents' decisions and the business environment as a bank president would, in clear graphic form. You get up-to-date economic information: T-bill and prime interest rates, consumer price indexes, GNP and stock market levels.

Your decisions lead your bank to success—or failure. Bank President uses sophisticated, realistic formulas to determine the economic consequences of your actions. To win, you must be a better Bank President than your opponents.

You use your computer for work. Now you can play, too—at your level. Contact Lewis Lee today to become—the Bank President.

LewisLee™ CORPORATION
1646 Portola Ave., Palo Alto, CA 94306 (415) 853-1220

Send to:	LEWIS LEE CORPORATION 1646 Portola Ave., Palo Alto, CA 94306
Phone orders:	(415) 853-1220

Please send me _______ copies of Bank President at \$74.95 each. (California residents add 6% sales tax.)

☐ IBM PC & XT ☐ Apple II, II+, IIe, III ☐ PC Jr.
☐ Lisa (February)
☐ Check or Money Order enclosed

☐ Visa ☐ MasterCard ☐ American Express

Name____________________
Address____________________
City____________ State______ Zip________
Card No.____________________
Exp. date____________________
Signature____________________
(Credit card charges not valid unless signed)
Dealer Inquiries Invited STI-1/84

struction about strategies for using it effectively. ETS feels that seven schools is about right for a network and that the coordinating agency should be an institution that has experience and expertise in teaching secondary-school teachers to use micros.

March 1983.

After much consideration, IBM makes the commitment to spend $8 million on its secondary-level computer education program. That amount will support a model program in three states. IBM selects New York, because it is corporate headquarters; Florida, because Boca Raton is the home of the PC, and California, because, as is the case with New York and Florida, IBM has substantial facilities there. Besides, the geographic distribution is good, and these three states account for nearly 20 percent of the nation's secondary school population.

IBM's senior management wants to ensure that the machines, which have not been purchased from local dealers, have technical service backup. Therefore, it's decided that within the three states the selected schools must be located near IBM's support facilities. Further, the corporation wants a cross section of schools—some public, some private, including parochial schools and schools for the handicapped. Ideally students will also cross economic, ethnic, and sociological lines.

The selected teacher training institutions should be those most qualified, judged on the basis of their experience; consequently, technical training centers as well as public and private universities will be included.

The $8 million, it is decided, will support twelve networks, approximately eighty-five schools in all. Each school will receive fifteen 128K PCs equipped with color monitors and double-sided disk drives. In addition, three printers will go to each school, along with software. And each teacher training institute will get a $10,000 cash grant, along with fifteen PCs and five printers.

So far, so good. ETS is confident the plan is sound, and IBM approves it.

Then comes the clincher. Unlike in education, where you plan for a year, "in the for-profit corporate world, when they decide they're going to do it, they want it done," according to Tony Cline. IBM wanted it done, all right. . .by the opening of school in September.

"This was the middle of March," Cline would later recall. "We were incredulous."

Six months

to deadline, and all ETS has is a plan. It does not have agreements with the necessary teacher training institutes, four in each of the three states. In fact, it doesn't even know which ones they should be. And, of course, the training institutes will represent more than eighty-five schools in the network plan, and ETS doesn't know which schools should be included either.

Then, of course, there is the training itself to be thought of. IBM will have to train ETS on the hardware; the group can figure out the software on its own. Next, ETS will have to train the teacher trainers on the PC, and these trainers will in turn have to set up programs for the high school teachers, some three to five from each school, or about three hundred teachers. All by September.

Moving ahead immediately, Cline calls the governors of the three target states, sets up meetings with the state superintendents of education, and looks for recommendations of teacher training institutes. Then come the visits to these institutes, along with lots of meetings. "Anyone," says Cline, "will have lunch with you if you've got fifteen PCs and $10,000 to give away."

Simultaneously, secondary schools are scouted, with the potential training institutes offering recommendations of schools known to have some experience with computers and located within the agreed-upon one-hundred-mile radius of IBM's specified support facilities. In California, these facilities are in Los Angeles and San Jose; in Florida, they're in Boca Raton and Tampa, and in New York, they're located in New York City and Westchester County.

May 1983.

ETS has determined the twelve teacher training institutes (TTIs) and arranged for two weeks' worth of intensive training for two instructors from each of them, beginning May 23 in Princeton. The TTIs chosen are a good representative cross section, ranging from the center at prestigious Vassar College in Poughkeepsie to the relatively unknown Polk County Schools teacher center in rural Florida.

One of California's teacher training institutes is at southern California's Pepperdine University. And within Pepperdine's potential network is El Camino Real High School. El Camino principal Larry Foster has gotten wind of the program and starts campaigning for his school's inclusion.

Diane Abrams doesn't know it, but even now she's closing in on learning to use a computer.

At Pepperdine are Terry Cannings, architect of the university's master of science degree in educational computing, and Jack McManus, director of computer services. Both hold doctoral degrees and have a good deal of experience in education. McManus taught French and Latin before succumbing to an advertisement designed to promote a program in computational linguistics. Cannings, an Australian who was once engaged in research for the down-under department of education, has been teaching for twenty years. The ad that hooked him on computers was one describing an educational computing conference sponsored by UCLA. "Ten people came—ten, to a national conference. Just shows what's happened since 1978."

May 22, 1983. With hardly time to rearrange their schedules and pack their bags, McManus and Cannings are off to Princeton as Pepperdine's teacher trainers. Both know computers, but neither knows the PC or its software. They'll have two weeks to learn everything from how to back up data to running graphics packages; at the same time, they'll also be attempting to determine the best ways to teach high school faculty members to use the machines. Most important, they'll be looking for answers to some bottom-line questions: How can *PFS:File* be used in the social sciences? Is a graphics package applicable to an art course? Can a program like *Delta Drawing* be an aid in problem solving? What would happen if word processing were incorporated into a creative writing class?

The twenty-four instructors, two from each of the twelve TTIs, meet at ETS's parklike setting. Accommodations are provided on-site; there's no time to commute from the Holiday Inn.

In charge of instruction is ETS's Martin Schneiderman, assisted by Roger Kershaw and Randy Bennett. Later on, when he's had a chance to catch his breath, Jack McManus will describe the instruction these trainers gave as "the most intensive and exciting training I've ever been involved in."

The teachers-turned-students begin at the beginning—unpacking and setting up—and soon discover that they have only fifteen to thirty minutes per software package, which must include opening it up, booting it, and coming up with instructional applications for it. Before the allotted time runs out, there will be visits from software vendors and from IBM, committee meetings on social and ethical issues, and gatherings of special interest groups. And somehow, in the midst of everything else, they'll have to find time to prepare outlines to use later in implementing the training sessions for the high school teachers.

Realizing that they are in for a four-week session crammed into two, McManus and Cannings devise a strategy, a way of handling the situation: One will listen to a vendor, the other will attend a committee meeting. One will observe or take part in one activity, the other will check out another. They'll compare notes later.

The machines are accessible twenty-four hours a day; not surprisingly, five or more people can be found working on them at two in the morning, at four, and at six. There is information overload. McManus: "By the third day, I was forgetting what I'd learned on the first—simple things, like how to boot a disk. I didn't know if I was overwhelmed or if I'd suddenly got dumb."

And then, it's over. The machines are packed up for the participants to take back with them; they'll have these to work with until each institution's shipment arrives. Everyone goes home.

It's now pushing the second week of June. On July 5, some twenty-eight teachers from Pepperdine's network of seven secondary schools will walk in the door, only about 20 percent of whom will have any computer science background. McManus and Cannings will go to work on their instruction plan. There will be no time for programming. ETS has insisted that the entire spectrum of secondary education be covered, not just math and science, so that means there'll be teachers of art, physics, English, business, and the social sciences.

Over at El Camino High School, Larry Foster is jubilant. He's just gotten the green light from a school board that had to find money in its beleaguered budget to pay unexpected four-week salaries to four of his teachers. Now, which four? When Foster announces the school's participation in the IBM program, twenty-five teachers ask to receive the training.

Meanwhile, McManus and Cannings are busy lining up speakers, checking out a videotape instruction package, refining how to approach the training of teachers who have no computer experience, and working out the details of ongoing support. Squeezed into this pressure-cooker schedule are inspections of school facilities. All schools are checked for electrical requirements, and each must provide its own security measures and furnishings—there's no IBM money for that. Southern California's Brentwood High School is in the process of building a million-dollar computer lab, but it won't be ready in time, so other facilities will have to be found instead. For security reasons, nearby Venice High School will have to cover a set of windows with bulletin boards; the first-floor location of its new computer room will be too risky otherwise. And since this room formerly served as a chemistry lab, McManus suggests that the water to the sinks be cut off.

Back at El Camino Real, Foster engages the support of the school's booster club in an effort to get additional funds. A parent provides, at cost, a sophisticated alarm system that detects body heat and vibration. And an enthusiastic Martha Baron, chairman of the business education department, learns that she is one of the privileged. "I didn't tell anyone, but I would have gone even if they hadn't paid me," Baron would admit later. Also chosen is Cay Gregory, head of the English department. Doris Dell and Pat Flenner, mathematics and computer science respectively, will split the four-week session, each getting two weeks apiece. Donnamae Huberman, science department chair, will go too.

One of Huberman's students is a bright, vivacious young woman who plans to be a surgeon: Diane Abrams.

July 5, 1983. The teachers arrive at Pepperdine. Some are excited, some apprehensive. McManus and Cannings feel both enthusiastic and anxious, but they're ready to begin, despite the unrealistic preparation schedule. "Oh," McManus would recall, "we were ready a good hour and a half before they got here."

The July session turns out to be Princeton revisited, but somewhat more humane by comparison. Organized activities are scheduled to happen between 9:30 and 4:00, and the room will be open from 8:30 to 5:00. In the beginning, teachers will work in pairs on computers that have already been set up; the other computers, all previously burned in by IBM, will be unpacked and set up by the teachers later on. The teachers from each school get a whole shelfful of software, including *Multiplan*, *PFS:File*, *PFS:Report*, *PFS:Graph*, *The Instruction*, *Question*, *Snooper Troops*, DOS 1.1, and *VisiCalc*.

The first two or three days, Cannings and McManus have to keep reminding themselves that what's now intuitively obvious to them—how to back up a disk, for example—is still a mystery to the teachers. But by the time the first day of the second week rolls 'round, McManus isn't sure that the teachers need them anymore. Cannings suggests they'd better stick around anyhow.

The teachers keep showing up early, so it's decided that the door to the computer lab should open at 8:00. Vendor presentations, special interest group and committee meetings, lesson planning, and faculty development planning must all be fit into four weeks. So must visits

(Left) Martha Baron, foreground, Donnamae Huberman, middle, and Cay Gregory with their newest teaching assistant. (Top) The Pepperdine graduating class—everyone, including McManus and Cannings, center in special T-shirts and caps, poses for the class picture. For McManus and Cannings the project has ended; for the new PC initiates, it's only begun.

from ETS, IBM, and two high school principals; two crossover meetings with the other southern California network at the Los Angeles TEC center; even a Friends' Day at which Martha Baron's tennis partners learn why she wasn't on the courts that summer. "I told them it's a real disease. And I love it."

By the end of the third week, the teachers have learned the hardware and the software. Now they begin talking about how to use the PC to teach everything from reading comprehension to art to trigonometry. A social studies teacher begins designing a database for her class so they can look at election projections for 1984 using demographics, issues, candidate statements, and polls.

PFS:File is a big hit, Logo gets mixed reviews, Cannings and McManus get raves. "They were fantastic," says Baron. "They were so intensely committed and so excited by what we were doing. It was the most exciting thing I've ever done educationally, and I've been teaching twenty-three years."

Echoing her is McManus, who observes, "These teachers will never be the same. It wasn't clear when we began that we could pull it off, because we were depending on teachers an awful lot to develop the instructional protocols and the content for the different subject matter areas. The program was extremely successful; it went beyond anything Terry and I dreamed we could do."

On the last day, there's a party and graduation. The Brentwood High teachers surprise their instructors with special T-shirts and caps. Plans are made for the monthly networking reunion, at which ideas, concerns, plans, and successes will be shared. Everyone poses for the class picture.

Midsummer, 1983. At the end of October, all the instructors from California, New York, and Florida will get together. But for now the frantic pace is over for McManus and Cannings. Cannings recuperates with a trip to Australia, while McManus, admitting to a somewhat empty feeling as a result of a diminished activity level, spends most his waking hours at the PC. "The phone is becoming an annoyance," he confides. "People who call me can hear the keys click in the background."

There's time enough for Pepperdine to schedule its fifteen PCs into its masters and doctoral classes. Once everything's arranged, McManus and Cannings will train Pepperdine's faculty and explore the possibility of holding Saturday sessions for members of the community. "We should show the parents what's going on in the schools," says McManus. "It's got to be a little bit frightening if your kid can run computers and you can't."

So, what began at an IBM board meeting has now moved into its ultimate arena: At some eighty-five secondary schools in three states, more than three hundred teachers are reviewing software, preparing lesson plans, making schedules for their PCs, and conferring with faculty members. The teachers who have been trained during the summer have the additional responsibility of training other teachers to use the PCs before the students of those teachers can use them.

Closed blinds conceal steel grids on the third-floor windows of the computer room at El Camino Real High School. All the PCs sport four-inch steel plates labeled "Hold-It," connected to steel cables attached to tables that are bolted to the floor. Concealed somewhere in the room is a blinking alarm system. Maintenance people know that they should mop, not sweep, the floor, and the chalkboard has been replaced with a board that accommodates a grease pencil. Principal Larry Foster has tried to think of everything; he's taking no chances with the fifteen precious PCs now that his school's got them.

Donnamae Huberman wears the key to the computer room on a length of bright pink yarn around her neck. Huberman, who admits to having been a "petrified klutz" before the training began, has designed a biorhythm research project for her premed students; through it, they'll learn to use *PFS:File*, *PFS:Report*, and *PFS:Graph*.

On this particular Friday early in the fall term, her class is learning to use *EasyWriter* to record notes. The students, at the level of a low roar, are exclaiming over successes and frustrations—and seeing immediate advantages to using the PC for other classes.

One of these students, determined to master word processing quickly, will spend her lunch hour in this room full of IBM's generosity. She's got an essay on Thomas Paine due Monday. ▲

INSTITUTES AND SCHOOLS

IBM/ETS Secondary School Computer Education Program

Teacher-Training Institutes — Network Schools

CALIFORNIA

Los Angeles Teacher Education Center, *Downey*

Banning High School, *Wilmington*
Bell Gardens High School
Bishop Montgomery High School, *Torrance*
Diamond Bar High School
Downey High School
Garfield High School, *Wilmington*
Manual Arts High School, *Los Angeles*
Valley High School, *La Puente*

Pepperdine University *Systems and Planning Division, Malibu*

The Brentwood School, *Los Angeles*
The Downtown Business Magnet High School, *Los Angeles*
El Camino Real Senior High School, *Woodland Hills*
Los Angeles Lutheran High School, *Burbank*
San Fernando Valley Christian High School, *Sepulveda*
Santa Monica High School
Venice Senior High School, *Los Angeles*

Santa Clara Teacher Education Computer Center, *San Jose*

Blackford High School, *San Jose*
Half Moon Bay High School
Los Altos High School
Menlo Atherton High School, *Atherton*
Santa Clara High School
Soquel High School

San Jose State University, *Department of Instructional Technology*

Bellarmine College Preparatory, *San Jose*
California School for the Deaf, *Fremont*
Gilroy High School
Institute of Computer Technology, *Sunnyvale*
William C. Overfelt High School, *San Jose*
Pioneer High School, *San Jose*
Santa Theresa High School, *San Jose*

FLORIDA

Polk County Schools, *Bartow*

Fort Meade Junior-Senior High School
Kathleen Senior High School, *Lakeland*
Lake Gibson Senior High School, *Lakeland*
Mulberry Junior-Senior High School
Winter Haven Senior High School

University of Southern Florida College of Education, *Tampa*

Boca Ciega High School, *St. Petersburg*
Chamberlain High School, *Tampa*
Countryside Senior High School, *Clearwater*
Hudson Senior High School
Plant City High School
Plant High School, *Tampa*
Sarasota Senior High School

Florida Atlantic University Educational Futures Lab, *Boca Raton*

Atlantic High School, *Delray Beach*
Boca Raton Academy
Boca Raton High School
Deerfield Beach High School
Lake Worth High School
Pope John Paul II High School, *Boca Raton*
The Potomac School, *Boca Raton*
J.P. Taravella High School, *Coral Springs*

Barry University Center for Computer Education, *Miami Shores*

American Senior High School, *Hialeah*
Archbishop Curley/Notre Dame High School, *Miami*
Boyd H. Anderson High School, *Lauderdale Lakes*
Coral Gables Senior High School
Greater Miami Hebrew Academy
Miami Edison Senior High School
The Pine Crest School, *Fort Lauderdale*
St. Thomas Aquinas High School, *Fort Lauderdale*
Miccosukee Indian School, *Miami*

NEW YORK

Queens College Office of Microcomputer Use School of Education, *Flushing*

Brooklyn Technical High School
John Dewey High School, *Brooklyn*
Lexington School for the Deaf, *Jackson Heights*
The Manhattan Center for Science and Mathematics, *New York*
Martin Luther King Senior High School, *New York*
Newtown High School, *Elmhurst*
Theodore Roosevelt High School, *Bronx*
The Ramaz School, *New York*
Susan E. Wagner High School, *Staten Island*

Putnam/Northern Westchester B.O.C.E.S., *Yorktown Heights*

Brewster High School, *Hopewell Junction*
John Jay Senior High School, *Katonah*
The Hackley School, *Tarrytown*
John F. Kennedy High School, *Somers*
Peekskill High School
Yorktown High School
White Plains High School

Vassar College Academic Program Development, *Poughkeepsie*

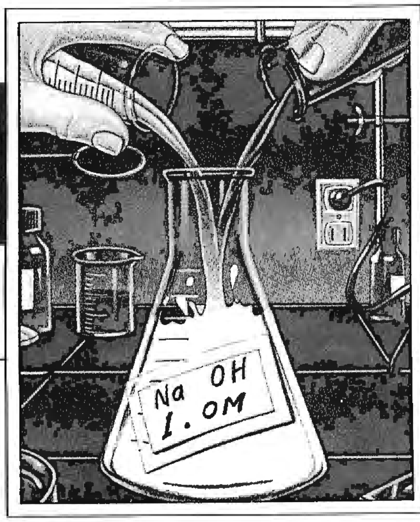
Arlington Senior High School, *Poughkeepsie*
Beacon Hill School
John Jay High School, *Hopewell Junction*
Kingston High School
Our Lady of Lourdes High School, *Poughkeepsie*
Poughkeepsie High School
Roy C. Ketcham High School, *Wappingers Falls*
Spackenkill High School, *Poughkeepsie*

Broome-Tioga B.O.C.E.S., *Binghamton*

Newark Valley Junior-Senior High School
Oneonta High School
Oswego Free Academy
Seton Catholic Central High, *Binghamton*
Tioga High School
Union Endicott High School, *Endicott*
Vestal High School

Information in chart courtesy of *PC Learning*, the newsletter of the IBM/ETS Secondary Computer Education Program, published by the Educational Testing Service, Princeton, NJ.

THE BASIC SOLUTION



by Joe Juhasz

A String Parser in Basic

Occasionally, when you're designing a computer program, you may want to be able to break down an entire line or sentence into individual pieces or words.

Certain adventure games provide a good example of this application. In these games, the player inputs a command by entering a sentence, such as *Open brick door with magic.* To make sense of this command, the program must first break the sentence down into the individual pieces or words: *Open*, *brick*, *door*, *with*, and *magic*.

This process is called *parsing*, and the program routine that does this work is called a *parser. The American Heritage Dictionary of the English Language* defines the verb *to parse* as: "to break (a sentence) down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part."

The Basic interpreter itself incorporates a parser that allows it to analyze statements such as: *A = B = C(I)* into their components and to determine whether or not these statements are syntactically correct.

This month's Basic subroutine, Parse, takes a statement (or sentence) and breaks it into its parts (words). For a parser to do its job, it must know what characters separate one word from another. In our example, commas and spaces function as separators. Parse also considers strings contained within single quotes to be a single component or word.

Here's an example:

PRINT 'Hi There', 10,5

breaks down to:

〈PRINT〉
〈"HI THERE"〉
〈10〉
〈5〉

Lines 1390 through 1430 need to be executed only once, during initialization. Notice that in lines 1440 and 1470 the Basic command *lset* is used. This is done to avoid string storage fragmentation (see "The Basic Solution," October and November 1983). Because of this approach, I$ and I1$ must *not* have their lengths changed elsewhere in the program. You may wish to rename these variables to prevent misuse. ▲

```
1010 '
1020 ' ***************************************************
1030 ' ***                                                      ***
1040 ' ***  Subroutine:   PARSE                                 ***
1050 ' ***  Language:     BASIC or BASICA                       ***
1060 ' ***  To Call:      GOSUB {line#}                         ***
1070 ' ***                                                      ***
1080 ' ***  Purpose:      PARSE takes an input string and breaks it into   ***
1090 ' ***                components (words). PARSE recognizes spaces      ***
1100 ' ***                and commas as separators. Strings within         ***
1110 ' ***                quote marks are considered as one word.          ***
1120 ' ***                                                      ***
1130 ' ***  Input:        IN$       — the string to be parsed            ***
1140 ' ***                                                      ***
1150 ' ***  Output:       WORDS     — the number of words found in INS   ***
1160 ' ***                WORDS$()  — an array containing the words found ***
1170 ' ***                                                      ***
1180 ' ***  Constants:    QUOTE$    — the character used as a quote      ***
1190 ' ***                TRUE      — logical variable, equates to true, -1  ***
1200 ' ***                FALSE     — logical variable, equates to false, 0  ***
1210 ' ***                                                      ***
1220 ' ***  Work Variables:                                       ***
1230 ' ***                I$        — remaining part of I$ (left justified)  ***
1240 ' ***                MAXL      — number of chars. in I$ left to parse   ***
1250 ' ***                I1$       — one-byte temporary variable        ***
1260 ' ***                SP        — temporary counter used to go through I$ ***
1270 ' ***                SP1       — location of separator #1 (space)   ***
1280 ' ***                SP2       — location of separator #2 (comma)   ***
1290 ' ***                D.Q       — logic variable, set if quote is found  ***
1300 ' ***                                                      ***
1310 ' ***************************************************
1320 '
1330 '
1340 '       ************************
1350 '       ***      PARSE I$     ***
1360 '       ************************
1370 '
1380 '—Initialize some variables
1390       QUOTE$=""                            '—Define character used as quote
1400       TRUE =-1                             '—Set up logical variable TRUE
1410       FALSE =0                             '—Set up logical variable FALSE
1420       I1$=SPACE$(1)                        '—Set I1$ to a length of one byte
1430       I$=SPACE$(255)                       '—Set I$ to a length of 255 bytes
1440 '—Parse Entry Point
1450       WORDS =0                             '—Set number of words found to 0
1460       LSET I$=IN$: MAXL=LEN(I$)            '—Set I$ and MAXL to start parse
1470    '**-get first nonspace-**
1480       FOR SP=1 TO MAXL
1490         LSET I1$=MID$(I$,SP,1)
1500         IF I1$<>SPACE$(1) GOTO 1600        '—Found nonspace, go process word
1510       NEXT
1520    '
1530    '**—finished parsing — found all words-**     *** EXIT ***
1540      PRINT "Found";WORDS;" words"
1550      FOR I=1 TO WORDS
1560        PRINT WORDS$(I)
1570      NEXT
1580 RETURN
1590    '**—find a word—**
1600     D.Q=(I1$=QUOTE$)                       '—If quote then set logical variable
1610     LSET I$=MID$(I$,SP): MAXL=MAXL-SP+1-D.Q
1620     IF NOT D.Q GOTO 1670                   '—If not quote then branch
1630      '*—string within quotes must be handled specially—*
1640      SP=INSTR(2,I$,QUOTE$)+1: IF SP=1 THEN SP=MAXL+1   '—Find end quote
1650      GOTO 1700
1660    '**—find first separator—*
1670     SP1=INSTR(I$," "): IF SP1=0 THEN SP1=MAXL+1   '—Find next space
1680     SP2=INSTR(I$,","): IF SP2=0 THEN SP2=MAXL+1   '—Find next comma
1690     IF SP1<SP2 THEN SP=SP1 ELSE SP=SP2
1700    '***—got a word—***
1710     WORDS=WORDS+1                          '—Increment # words found
1720     WORDS$(WORDS) = LEFT$(I$,SP-1)         '—Store word in array
1730     LSET I$=MID$(I$,SP): MAXL=MAXL-SP+D.Q  '—Get remainder to parse
1740     D.Q=FALSE                              '—Reset D.Q for next word
1750 GOTO 1480                                  '—Try to get another word
1760 '
```

BLUE

COMES IN MANY COLORS

BLUE — THE WORD PROCESSOR
THE CHOICE ISN'T BLACK AND WHITE ANYMORE.

PRICE: $150.00

DEMO DISK: $10.00

WORKS GREAT IN MONOCHROME TOO!

DEALER INQUIRIES INVITED

IBM PERSONAL COMPUTER OR XT.
64K AND ONE DISK DRIVE.
DOS 1.1 OR DOS 2.0
MONOCHROME OR COLOR
GRAPHICS ADAPTER

SYMMETRIC SOFTWARE

3812 EAST LA PALMA AVENUE
ANAHEIM, CALIFORNIA 92807
(714) 630-8733

The Case of the Missing Files

Socha's Toolbox by John Socha

Hard disks are marvelously fast, and they make it easy for you to keep all your work in one place. But beware. Files on hard disks have a way of slipping mysteriously into the night. When you know that program you want is in some subdirectory somewhere, but you can't for the life of you remember where, what do you do? Search through all directories? Unless you get a sudden flash of inspiration, you don't have much choice. What you need is a program to do the searching for you.

And that is precisely what Whereis.com does; it scans all directories on a disk in search of missing files. In this article, we'll build Whereis.com using a Basic program to construct the .com file, so you won't need any knowledge of machine language.

Report of Missing Files. Let's say we have a disk with a total of five directories, as shown in figure 1, and we've forgotten that we put Lostfile.txt in \data\monthly. How might we find this file?

One way would be to use the DOS 2.0 *tree/f* command. That's not much fun, however, since *tree/f* forces us to watch our entire list of files scroll past while we hunt for the one that's missing. Still, the method does work.

Suppose we need to find all files that start with "lost"? That's a bit harder.

DOS's *dir* command—in the form *dir lost *.** —would locate all files starting with "lost," provided that we have only one directory. But, of course, that's not the case. DOS 2.0 lets us have any number of directories.

So why didn't IBM provide a directory command that would search through *all* directories? They should have. Well, since they didn't put this feature into the *dir* command (nor is it supplied anywhere else), we'll just have to do it ourselves.

Once you've built Whereis.com, you'll be able to type *whereis lost *.** and see the directory locations for all files starting with "lost." For the example shown in figure 1, Whereis will return:

\DATA\LOSTNOT.TXT
\DATA\MONTHLY\LOSTFILE.TXT

Building the Case. Here's the Basic program that builds Whereis.com. Enter this program and run it once to create the .com file on your disk. If you mistype any of the data lines, the Basic program will tell you which lines are wrong.

```
10 DIM CHECK(39)
20 FOR I=1 TO 39 : CHECK(I)=0 : NEXT I
30 PRINT "Checking";
40 FOR I=1 TO 39
50    FOR J=1 TO 8
60       READ BYTE
70       CHECK(I) = CHECK(I) XOR BYTE
80    NEXT J
90    PRINT ".";
100 NEXT I
110 PRINT
120 LINECHECK=0
130 FOR I=1 TO 39
140    READ CHECK
150    LINECHECK = LINECHECK XOR CHECK
160    IF CHECK(I) <> CHECK THEN PRINT "Data in Line";1000+10*(I-1);"may be bad."
170 NEXT I
180 IF LINECHECK <> 230 THEN PRINT "Data bad in lines 2010-2050."
190 OPEN "whereis.com" AS #1 LEN=1
200 PRINT "Writing. . ."
210 FIELD #1,1 AS BYTES
220 RESTORE
230 FOR I=1 TO 312        'Read the code
240    READ BYTE : LSET BYTES = CHRS(BYTE) : PUT #1
250 NEXT I
260 LSET BYTES = CHRS(0)
270 FOR I=1 TO 91         '91 bytes of 0 from data area
280    PUT #1
290 NEXT I
300 CLOSE
310 PRINT "WHEREIS.COM created"
320 END
1000 DATA  190,  130,  0,    191,  134,  2,    172,  60
1010 DATA  13,   116,  3,    170,  235,  248,  50,   192
1020 DATA  170,  191,  52,   2,    50,   192,  252,  185
1030 DATA  64,   0,    242,  174,  139,  223,  75,   186
1040 DATA  0,    0,    232,  2,    0,    205,  32,   86
1050 DATA  82,   232,  132,  0,    232,  161,  0,    114
1060 DATA  13,   232,  95,   0,    232,  183,  0,    114
1070 DATA  5,    232,  87,   0,    235,  246,  90,   82
1080 DATA  232,  118,  0,    232,  138,  0,    114,  36
1090 DATA  139,  242,  246,  68,   21,   16,   117,  11
1100 DATA  232,  155,  0,    114,  23,   246,  68,   21
1110 DATA  16,   116,  245,  128,  124,  30,   46,   116
1120 DATA  239,  232,  11,   0,    80,   180,  26,   205
1130 DATA  33,   88,   235,  228,  90,   94,   195,  87
1140 DATA  86,   80,   83,   252,  139,  242,  131,  198
1150 DATA  30,   139,  251,  172,  170,  10,   192,  117
1160 DATA  250,  139,  223,  253,  170,  176,  92,   170
1170 DATA  232,  156,  255,  91,   198,  7,    0,    88
1180 DATA  94,   95,   195,  80,   82,   186,  52,   2
1190 DATA  138,  7,    198,  7,    0,    232,  108,  0
1200 DATA  136,  7,    90,   82,   131,  194,  30,   232
1210 DATA  98,   0,    232,  80,   0,    90,   88,   195
1220 DATA  86,   190,  134,  2,    232,  11,   0,    94
1230 DATA  195,  86,   190,  48,   2,    232,  2,    0
1240 DATA  94,   195,  80,   87,   139,  251,  252,  172
1250 DATA  170,  10,   192,  117,  250,  95,   88,   195
1260 DATA  81,   131,  250,  0,    119,  3,    186,  104
1270 DATA  2,    131,  194,  43,   185,  16,   0,    180
1280 DATA  26,   205,  33,   82,   186,  52,   2,    180
1290 DATA  78,   205,  33,   90,   89,   195,  81,   82
1300 DATA  186,  52,   2,    185,  16,   0,    180,  79
1310 DATA  205,  33,   90,   89,   195,  80,   82,   180
```

Help Yourself

MASTER CONTROL
A DIET & EXERCISE PROGRAM

A physician-designed, comprehensive weight management program.

Guiding you to a new eating and exercise lifestyle.

Allowing you to monitor your food intake and energy expenditure instantaneously.

Graphic reports provide feedback on your long range weight loss success.

Valuable programmed instruction in exercise and nutrition and other related topics.

A complete diet guide that will show you how to lose weight without relying on willpower.

·HEALTHWARE·

Dealer inquiries invited

Introductory Offer 69.95

Mail to:
Healthware/1504 Leander Rd./Georgetown, TX 78626/(512) 863-6910
Enclose $69.95 for each package and $3.00 shipping and handling.

☐ Send me ______ copy(s) of the Master Control Diet and Exercise Program.
☐ Check or money order enclosed
☐ MasterCard ☐ VISA

Name: ______________________

Address: ______________________

City: __________ State: __________ Zip: __________

Card No.: ______________________

Expiration Date: __________ Bank No. (if MC): __________

Signature: ______________________

MONEY-BACK GUARANTEE!
Return program package within 30 days if not completely satisfied and receive full refund.

```
1320 DATA  2,    178,  10,   205,  33,   178,  13,   205
1330 DATA  33,   90,   88,   195,  80,   82,   86,   252
1340 DATA  139,  242,  180,  2,    172,  138,  208,  205
1350 DATA  33,   172,  10,   192,  117,  247,  94,   90
1360 DATA  88,   195,  0,    0,    0,    0,    0,    0
1370 DATA  0,    0,    0,    0,    0,    0,    0,    0
1380 DATA  42,   46,   42,   0,    92,   0,    0,    0
1390 '91 bytes of 0s
2000 '
2010 DATA  151,  49,   148,  185,  81,   5,    151,  175
2020 DATA  170,  176,  177,  41,   63,   230,  149,  215
2030 DATA  191,  73,   76,   200,  48,   27,   209,  243
2040 DATA  186,  43,   142,  117,  156,  97,   222,  154
2050 DATA  36,   72,   244,  193,  155,  0,    114
```

Listing 1.

The Solution. That's all there is to Whereis. Try typing *whereis where*.** to see what happens. The program should turn up Whereis.bas and Whereis.com—and perhaps a few other files that start with "where."

Whereis uses a technique known as recursion. Here's how it works. The program starts by looking in the root directory, \ . Every time it finds a match, Whereis displays the full name of the matching file—the pathname plus the filename. In the root directory the pathname is just \ . Once Whereis has checked all files in the root directory, it checks each subdirectory in turn until it's gone through all subdirectories.

To see how Whereis goes about searching systematically through all subdirectories, let's look at a pseudocode description of the program.

Pseudocode is an English-language representation of a program. *File_name* is the name of one file in a directory, *search_name* is the name we're searching for, and *directory_name* is the name of the directory we're currently searching for. Here's the pseudocode for Whereis.com:

```
program SEARCH(search_name)
        SEARCH_DIRECTORY(\)
    end program
    subroutine SEARCH_DIRECTORY(directory_name):
        For all file_name in directory_name:
            If file_name matches search_name
                Print path_name\file_name
        For each sub_directory in directory_name:
            SEARCH_DIRECTORY(sub_directory)
        Return to previous SEARCH level
    end subroutine.
```

You may have noticed something unusual about this program: The procedure Search_directory calls itself. That's recursion. Search_directory searches one directory, then calls itself again to search each subdirectory. To see how the Search_directory subroutine works, we'll trace through a search for *where**. * in the directory and file structure shown in figure 1.

Search_directory starts looking for *where**. * in directory \ but finds no matches. The next instruction causes the subroutine to search through subdirectories. Now comes the sticky part.

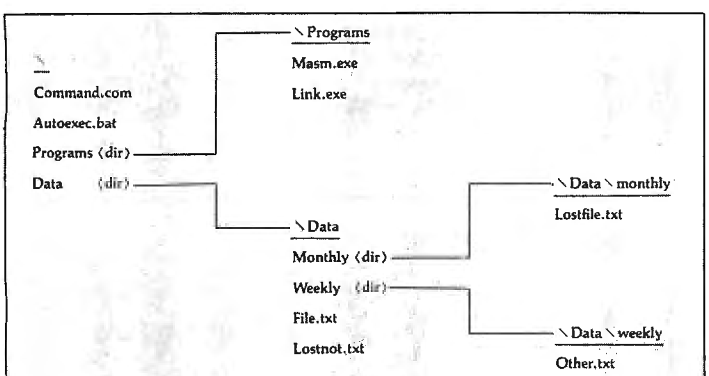


Figure 1.

The first subdirectory of \ is Programs. Search_directory(sub_directory) starts Search_directory again but uses the pathname of directory\sub_directory—which, in this case, is \Programs. Whereis remembers that it was doing a Search_directory before and that it's now on the second level of Search_directory. So when it's done searching \Programs, it'll move on with its search through \ . We'll come back to this point.

Whereis doesn't find *where**. * in \Programs, so the second level of Search_directory checks for subdirectories. But \Programs doesn't have any subdirectories. Therefore, Search_directory returns to the previous level of Search_directory—that is, it goes back to searching through \ , the root directory.

Back at \ , our subroutine has just finished looking through the Programs subdirectory. The next subdirectory in line is Data. So we start another level of Search_directory for \Data. This level of search finds Lostnot.txt and prints

\DATA\LOSTNOT.TXT

Hold on, we're not done yet. This second level of Search_directory still has to look through subdirectories. So it starts with Monthly (or \Data\Monthly). Now on the third level, Search_directory finds Lostfile.txt and prints

\DATA\MONTHLY\LOSTFILE.TXT

but it finds no subdirectories here, so it now backs up a level. Resuming its second-level search, Search_directory then finds the subdirectory \Data\Weekly and descends once more to a third-level search. For the structure shown in figure 1, that's the end of the line.

Whew.

Telling It Like It Is. Recursion makes very compact and reliable programs, but it's not as simple to write recursive programs in assembly language as to write them in languages like Pascal. Basic is even worse than assembly language for recursive programs. We can write recursive programs in assembly language by creating local variables; that is, we can save registers and variables at the start of a procedure and restore them just before we leave.

Whereis uses DOS 2.0 function 4EH to find the first match in a directory. It also calls upon function 1AH to tell DOS which memory to use for the Disk Transfer Area (DTA).

The DTA contains information that lets DOS know what file it's just found and where to search for the next match. Among other things, the DTA contains the pathname of a directory. Whereis starts its search by calling SEARCH_DIRECTORY; it descends (moves down the tree) by calling SEARCH_SUB_DIRECTORY, and it ascends (moves back up the tree) by calling SEARCH_DIRECTORY again. By using a new DTA each time SEARCH_DIRECTORY calls SEARCH_SUB_DIRECTORY, and by restoring the old DTA for each return from SEARCH_SUB_DIRECTORY, the program can keep track of its progress through subdirectories.

You'll find the details in listing 2.

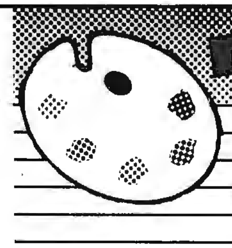
```
CGROUP    GROUP     CODE_SEG, DATA_SEG
          ASSUME    CS:CGROUP, DS:CGROUP

;------------------------------------------------------------------------;
; This is the format for the DOS Data Transfer Area used when DOS 2.0     ;
; searches for a file match in directories.                               ;
;------------------------------------------------------------------------;
DTA       STRUC
          RESERVED      DB     21 DUP (?)
          ATTRIBUTE     DB
          TIME          DW
          DATE          DW
          SIZE          DD
          NAME_FOUND    DB     13 DUP (?)
DTA       ENDS

DATA_SEG      SEGMENT  PUBLIC
STAR_NAME     DB       '*.*',0
PATH_NAME     DB       '\',0
```

FUN 'N' GAMES

GET 1 FREE WHEN YOU BUY 2 OR MORE!

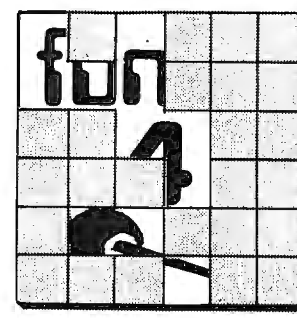
These well-designed programs are educational for kids and fun for the entire family. For a limited time, we're offering you 1 free when you buy two or more of equal or greater value. All provide hours of enjoyment, and make great Christmas gifts too!

PC CRAYON



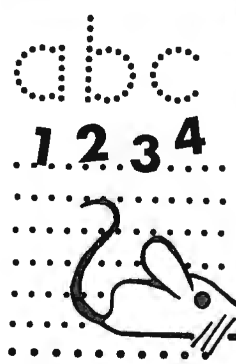
Is there an artist lurking inside you longing to create colorful pictures, animated displays and eyecatching slide shows? Let PCcrayon assist you! Anyone from six to sixty can quickly learn PCcrayon's powerful commands. And businesses can utilize "the Crayon" to create impressive presentations. Creations can be saved and re-used by your own programs. PCcrayon even creates hardcopy! Requires 64K, graphics adapter and display. **$44.95**

CONCENTRATE



Just try and see how accurate your memory really is! Based on the popular TV game show Concentration, players try to match squares and correctly guess the answer to colorful graphic puzzles. Educational for kids, and just plain fun for adults. Requires 128K, graphics adapter and display. **$44.95**

MY LETTERS, NUMBERS & WORDS

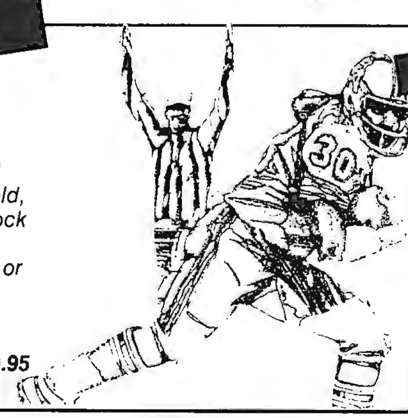


A pre-primary education software package for children from one to six. Children are taught the alphabet, the numbers from one to ten, and the concept of words through animated flash card routines and graphic displays. The program's attractive sounds and animated graphics will keep your children's attention and interest while they learn. Requires 64K, graphics adapter & display. **$39.95**

CHAMPIONSHIP BLACKJACK

Learn to win at the blackjack tables! You can improve your play with **Championship Blackjack**. *Perhaps the best blackjack game on the market,* **Championship Blackjack** *features realistic color display with sounds, casino rules you can set, "pro" playing strategies, up to six players at a table with all cards always in view, "double exposure" and more! Requires 64K, any adapter and display.* **$39.95**

ARMCHAIR QUARTERBACK



You're the quarterback! Lead your team to end-zone victory! A football strategy game including full representation of playing field, scoreboard with running clock and team statistics and overtime play. Play a friend or play your PC. No manual dexterity required. Requires 64K, any adapter & display. **$39.95**

CHESS 2001

The perfect chess partner! An elaborately designed program which offers detailed chess pieces represented on a chess board, six skill levels, no commands to memorize (the cursor keys control your moves!) Perfect for beginning chess players and excellent practice for intermediate players. Requires 64K, graphics adapter & display. **$39.95**

ORDER TODAY FOR YOUR FREE PROGRAM!

Call us at (619) 571-0981 and charge it to your Visa or MasterCharge card.

PCsoftware™
OF SAN DIEGO

9120 Gramercy Drive, Suite 416, San Diego, CA 92123 • Telephone (619) 571-0981

```
            DB      80 DUP (0)         ;Space for 64 character pathname
                                       ; and 13-character filename
FILE_NAME   DB      13 DUP (0)         ;Save room for full DOS filename
DISK_TRANSFER_AREAS LABEL BYTE         ;This starts at the end of the WHEREIS
DATA_SEG    ENDS

;-----------------------------------------------------------------------;
; This is the main program that sets up the initial conditions for     ;
; SEARCH_DIRECTORY which, in turn, does a recursive search.             ;
;                                                                       ;
; Reads:      PATH_NAME                                                 ;
; Writes:     FILE_NAME                                                 ;
; Calls:      SEARCH_DIRECTORY                                          ;
;-----------------------------------------------------------------------;
CODE_SEG    SEGMENT
      ORG   100H
WHEREIS     PROC   NEAR
      MOV   SI,82H                     ;Start of command line
      MOV   DI,OFFSET CGROUP:FILE_NAME
GET_SEARCH_NAME:
      LODSB                            ;Get first character
      CMP   AL,0DH                     ;Carriage return character?
      JE    DONE_READING_NAME          ;Yes, end of name
      STOSB
      JMP   GET_SEARCH_NAME
DONE_READING_NAME:
      XOR   AL,AL                      ;Write a 0 at the end
      STOSB
      MOV   DI,OFFSET CGROUP:PATH_NAME
      XOR   AL,AL                      ;Search for the zero at the end
      CLD                              ; of PATH_NAME
      MOV   CX,64                      ;Max. length of scan for 0
      REPNZ SCASB
      MOV   BX,DI
      DEC   BX                         ;DS:BX points to end of PATH_NAME
      MOV   DX,0                       ;Tell SEARCH_DIRECTORY this is first
      CALL  SEARCH_DIRECTORY           ;Now do the recursive search
      INT   20H                        ;All done now.
WHEREIS     ENDP

;-----------------------------------------------------------------------;
; This procedure searches all the files in the current directory       ;
; looking for a match. It also prints the full name for each match.    ;
;                                                                       ;
;       DS:BX      Pointer to end of current path name                  ;
;       DS:DX      Old disk transfer area (DTA)                         ;
;                                                                       ;
; Reads:      Disk Transfer Area (DTA)                                  ;
; Writes:     Disk Transfer Area (DTA)                                  ;
; Calls:      BUILD_NAME, GET_FIRST_MATCH,                              ;
;             WRITE_MATCHED_NAME, GET_NEXT_MATCH,                       ;
;             BUILD_STAR_NAME, SEARCH_SUB_DIRECTORY                     ;
;-----------------------------------------------------------------------;
SEARCH_DIRECTORY        PROC   NEAR
      PUSH  SI                         ;Need to restore on exit
      PUSH  DX
      CALL  BUILD_NAME                 ;Build the absolute search name
      CALL  GET_FIRST_MATCH            ;See if there is a match here
      JC    NO_MATCH                   ;No match, check subdirectories
      CALL  WRITE_MATCHED_NAME         ;Write name of match.
FIND_NEXT_FILE:
      CALL  GET_NEXT_MATCH             ;Find the next match
      JC    NO_MATCH                   ;No match, search subdirectories
      CALL  WRITE_MATCHED_NAME         ;Match, write absolute name
      JMP   FIND_NEXT_FILE             ;Look for the next matching name
NO_MATCH:                              ;No match, so try subdirectories.
      POP   DX                         ;Restore DTA
      PUSH  DX
      CALL  BUILD_STAR_NAME            ;Search for all directories
      CALL  GET_FIRST_MATCH            ;Get first entry
      JC    NO_MORE_MATCHES            ;There are no entries
      MOV   SI,DX                      ;Put address of DTA into SI
      TEST  [SI].ATTRIBUTE,10H         ;Is it a directory entry?
      JNZ   IS_DIRECTORY               ;Yes, then search subdirectory
FIND_NEXT_DIRECTORY:
      CALL  GET_NEXT_MATCH             ;No, then find the next match
      JC    NO_MORE_MATCHES            ;There are no more entries
      TEST  [SI].ATTRIBUTE,10H         ;Is this a directory?
      JZ    FIND_NEXT_DIRECTORY        ;No, then try again
IS_DIRECTORY:
      CMP   [SI].NAME_FOUND,'.'        ;Is this a . or .. directory?
      JE    FIND_NEXT_DIRECTORY        ;Yes, skip to next directory
      CALL  SEARCH_SUB_DIRECTORY       ;Search the subdirectory
      PUSH  AX                         ;Now reset the DTA
      MOV   AH,1AH
      INT   21H
      POP   AX
      JMP   FIND_NEXT_DIRECTORY
NO_MORE_MATCHES:
      POP   DX
      POP   SI
      RET
SEARCH_DIRECTORY        ENDP

;-----------------------------------------------------------------------;
; This procedure searches the subdirectory whose name is in the DTA    ;
;                                                                       ;
;       DS:BX             End of the current pathname                   ;
;       DS:[DX].NAME_FOUND     Name of subdirectory for search          ;
; Reads:      PATH_NAME                                                 ;
; Writes:     PATH_NAME                                                 ;
; Calls:      SEARCH_DIRECTORY                                          ;
;-----------------------------------------------------------------------;
SEARCH_SUB_DIRECTORY    PROC   NEAR
      PUSH  DI
      PUSH  SI
      PUSH  AX
      PUSH  BX
      CLD                              ;Set for increment
      MOV   SI,DX                      ;Put address of DTA into SI
      ADD   SI,OFFSET NAME_FOUND       ;Set to start of subdirectory name
      MOV   DI,BX                      ;DS:DI -- 0 at end of pathname
COPY_LOOP:                             ;Copy subdirectory to pathname
      LODSB                            ;Copy one character
      STOSB
      OR    AL,AL                      ;Was it a 0?
      JNZ   COPY_LOOP                  ;No, keep copying
      MOV   BX,DI                      ;Set BX to end of new pathname
      STD                              ;Set flag for decrement
      STOSB                            ;Store a 0 at end of string
      MOV   AL,'\'
      STOSB                            ;Place '\' at end of pathname
      CALL  SEARCH_DIRECTORY           ;Search this new path
      POP   BX                         ;Restore the old end-of-path
      MOV   BYTE PTR [BX],0            ;And store a zero here
      POP   AX
      POP   SI
      POP   DI
      RET
SEARCH_SUB_DIRECTORY    ENDP
```

```
;-------------------------------------------------------------------------;
;  This procedure prints the matched name after the path name              ;
;                                                                          ;
;       DS:DX          Pointer to current disk transfer area               ;
;                                                                          ;
; Reads:        PATH_NAME, NAME_FOUND (in DTA)                             ;
; Calls:        WRITE_STRING, SEND_CRLF                                    ;
;-------------------------------------------------------------------------;
WRITE_MATCHED_NAME    PROC    NEAR
        PUSH    AX
        PUSH    DX
        MOV     DX,OFFSET CGROUP:PATH_NAME
        MOV     AL,[BX]                         ;Save character at end of path
        MOV     BYTE PTR [BX],0                 ;Set for end of string
        CALL    WRITE_STRING
        MOV     [BX],AL                         ;Restore character
        POP     DX                              ;Recover old pointer
        PUSH    DX
        ADD     DX,OFFSET NAME_FOUND
        CALL    WRITE_STRING
        CALL    SEND_CRLF
        POP     DX
        POP     AX
        RET
WRITE_MATCHED_NAME    ENDP
;-------------------------------------------------------------------------;
;  This procedure builds an absolute search name from PATH_NAME            ;
; followed by FILE_NAME.                                                   ;
;                                                                          ;
; Reads:        FILE_NAME                                                  ;
; Calls:        BUILD         to build the name                            ;
;-------------------------------------------------------------------------;
BUILD_NAME  PROC        NEAR
        PUSH    SI
        MOV     SI,OFFSET CGROUP:FILE_NAME
        CALL    BUILD
        POP     SI
        RET
BUILD_NAME  ENDP
BUILD_STAR_NAME         PROC    NEAR
        PUSH    SI
        MOV     SI,OFFSET CGROUP:STAR_NAME
        CALL    BUILD
        POP     SI
        RET
BUILD_STAR_NAME         ENDP
;-------------------------------------------------------------------------;
;  This procedure appends the string at DS:SI to PATH_NAME in              ;
; PATH_NAME. It knows where the path name ends from knowing how            ;
; long PATH_NAME is.                                                       ;
;                                                                          ;
;        DS:SI          Name of file                                       ;
;        DS:BX          End of PATH_NAME                                   ;
;                                                                          ;
; Reads:        DS:SI                                                      ;
; Writes:       PATH_NAME                                                  ;
;-------------------------------------------------------------------------;
BUILD   PROC    NEAR
        PUSH    AX
        PUSH    DI
        MOV     DI,BX
        CLD                                     ;Set direction for increment
COPY_NAME:
        LODSB                                   ;Copy one character of name
        STOSB
        OR      AL,AL                           ;End of string yet?
        JNZ     COPY_NAME                       ;No, keep copying
        POP     DI
        POP     AX
        RET
BUILD   ENDP
;-------------------------------------------------------------------------;
;  This procedure finds the first match between the name given by          ;
; DS:DX and the directory entries found in the directory PATH_NAME.        ;
;                                                                          ;
;       DS:DX           Pointer to current disk transfer area              ;
; Returns:                                                                 ;
;       CF      0       A match was found                                  ;
;               1       No match found                                     ;
;       AX              Error code returned:                               ;
;               2       File not found                                     ;
;               18      No more files                                      ;
;       DS:DX           Pointer to new disk transfer area                  ;
;                                                                          ;
; Reads:        PATH_NAME                                                  ;
; Writes:       DISK_TRANSFER_AREAS                                        ;
;-------------------------------------------------------------------------;
```

```
GET_FIRST_MATCH         PROC    NEAR
        PUSH    CX
        CMP     DX,0                            ;First one?
        JA      ALLOCATE                        ;No, then allocate space
        MOV     DX,OFFSET CGROUP:DISK_TRANSFER_AREAS-TYPE DTA
ALLOCATE:
        ADD     DX,TYPE DTA                     ;No, then allocate room for new DTA
        MOV     CX,10H                          ;Search attribute for files and dirs.
        MOV     AH,1AH                          ;Set disk transfer address
        INT     21H
        PUSH    DX                              ;Need DX for address of search name
        MOV     DX,OFFSET CGROUP:PATH_NAME
        MOV     AH,4EH                          ;Call for "find first match"
        INT     21H
        POP     DX
        POP     CX
        RET                                     ;Return with carry flag info
GET_FIRST_MATCH         ENDP
;-------------------------------------------------------------------------;
;  This procedure is much like GET_FIRST_MATCH.                            ;
;                                                                          ;
; Returns:                                                                 ;
;       CF      0       A match was found                                  ;
;               1       No match found                                     ;
;       AX              Error code returned:                               ;
;               2       File not found                                     ;
;               18      No more files                                      ;
;                                                                          ;
; Reads:        PATH_NAME                                                  ;
; Writes:       DISK_TRANSFER_AREAS                                        ;
;-------------------------------------------------------------------------;
GET_NEXT_MATCH          PROC    NEAR
        PUSH    CX
        PUSH    DX
        MOV     DX,OFFSET CGROUP:PATH_NAME
        MOV     CX,10H                          ;Attribute for files and directories
        MOV     AH,4FH                          ;Call for "Find next match"
        INT     21H
        POP     DX
        POP     CX
        RET                                     ;Return with carry flag intact
GET_NEXT_MATCH          ENDP
;-------------------------------------------------------------------------;
;  This procedure sends a CRLF pair of characters to the screen.           ;
;-------------------------------------------------------------------------;
SEND_CRLF       PROC    NEAR
        PUSH    AX
        PUSH    DX
        MOV     AH,02
        MOV     DL,0AH
        INT     21H
        MOV     DL,0DH
        INT     21H
        POP     DX
        POP     AX
        RET
SEND_CRLF       ENDP
;-------------------------------------------------------------------------;
;  This procedure writes the ASCIIZ string at                              ;
;       DS:DX           Address of ASCIIZ string.                          ;
;-------------------------------------------------------------------------;
WRITE_STRING    PROC    NEAR
        PUSH    AX
        PUSH    DX
        PUSH    SI
        CLD                                     ;Set direction for increment
        MOV     SI,DX                           ;Set up pointer to string
        MOV     AH,2
        LODSB                                   ;Get the first character
WRITE_STRING_LOOP:
        MOV     DL,AL
        INT     21H                             ;Write one character
        LODSB
        OR      AL,AL                           ;End of the string?
        JNZ     WRITE_STRING_LOOP
        POP     SI
        POP     DX
        POP     AX
        RET
WRITE_STRING    ENDP

CODE_SEG        ENDS

        END     WHEREIS
```

Listing 2.

Of Bugs and Men

There comes a time in the lives of mice and men when the sky turns dark and the bugs attack. My program Kbd_fix.com, presented in the November *Softalk* ("Enhance Your Keyboard Buffer," by John Socha, page 117), contains a serious error. To make matters worse, there's also an error in the Basic program that builds Kbd_fix.com. It was a bad month.

The Locust Swarm. Kbd_fix.com works perfectly, provided you're not running a program that uses INT 16H function 2—the ROM BIOS routine to check the status of the various shift keys. If your program should call that status check function, however, that does it—your machine will probably head straight toward limbo. And your only recourse will be to turn off the machine for a few seconds and start over.

```
180 IF LINECHECK <> 59 THEN PRINT "Data bad in lines 2010-2060."
1000 DATA 233,  88,  2,   0,   0,   0,   0,   0
1010 DATA 0,    0,   0,   15,  1,   15,  1,   0
1015 '312 bytes of 0s
1020 DATA 0,    0,   0,   0,   0,   0,   0,   80
1030 DATA 83,   81,  156, 250, 187, 30,  0,   228
1040 DATA 97,   80,  36,  252, 230, 97,  185, 60
1050 DATA 0,    226, 254, 12,  2,   230, 97,  185
1060 DATA 60,   0,   226, 254, 75,  117, 235, 88
1070 DATA 230,  97,  157, 89,  91,  88,  195, 187
1080 DATA 64,   0,   142, 219, 250, 139, 30,  26
1090 DATA 0,    59,  30,  28,  0,   117, 29,  187
1100 DATA 30,   0,   137, 30,  26,  0,   131, 195
1110 DATA 2,    137, 30,  28,  0,   140, 203, 142
1120 DATA 219,  187, 15,  1,   137, 30,  11,  1
1130 DATA 137,  30,  13,  1,   251, 195, 30,  86
1140 DATA 83,   80,  232, 202, 255, 156, 46,  255
1150 DATA 30,   3,   1,   187, 64,  0,   142, 219
1160 DATA 46,   139, 54,  13,  1,   139, 30,  26
1170 DATA 0,    131, 195, 2,   131, 251, 62,  114
1180 DATA 3,    187, 30,  0,   59,  30,  28,  0
1190 DATA 116,  38,  139, 7,   46,  137, 4,   131
1200 DATA 198,  2,   129, 254, 79,  2,   114, 3
1210 DATA 190,  15,  1,   46,  59,  54,  11,  1
1220 DATA 117,  5,   232, 98,  255, 235, 5,   46
1230 DATA 137,  54,  13,  1,   137, 30,  26,  0
1240 DATA 160,  23,  0,   36,  12,  60,  12,  117
1250 DATA 8,    46,  161, 13,  1,   46,  163, 11
1260 DATA 1,    88,  91,  94,  31,  207, 251, 30
1270 DATA 83,   232, 99,  255, 140, 203, 142, 219
1280 DATA 10,   228, 116, 12,  128, 252, 1,   116
1290 DATA 41,   91,  31,  46,  255, 46,  7,   1
1300 DATA 251,  144, 250, 139, 30,  11,  1,   59
1310 DATA 30,   13,  1,   116, 243, 139, 7,   131
1320 DATA 195,  2,   129, 251, 79,  2,   117, 3
1330 DATA 187,  15,  1,   137, 30,  11,  1,   91
1340 DATA 31,   207, 250, 139, 30,  11,  1,   59
1350 DATA 30,   13,  1,   139, 7,   251, 91,  31
1360 DATA 202,  2,   0,   30,  184, 0,   0,   142
1370 DATA 216,  250, 161, 36,  0,   46,  163, 3
1380 DATA 1,    161, 38,  0,   46,  163, 5,   1
1390 DATA 199,  6,   36,  0,   166, 2,   140, 14
1400 DATA 38,   0,   251, 161, 88,  0,   46,  163
1410 DATA 7,    1,   161, 90,  0,   46,  163, 9
1420 DATA 1,    199, 6,   88,  0,   14,  3,   140
1430 DATA 14,   90,  0,   184, 64,  0,   142, 216
1440 DATA 250,  187, 30,  0,   137, 30,  26,  0
1450 DATA 199,  7,   0,   0,   131, 195, 2,   137
1460 DATA 30,   28,  0,   251, 186, 91,  3,   205
1470 DATA 39,   0,   0,   0,   0,   0,   0,   0
2000 '
2010 DATA 179,  0,   80,  37,  235, 44,  173, 56
2020 DATA 96,   234, 211, 64,  243, 235, 147, 178
2030 DATA 16,   118, 159, 254, 135, 153, 197, 62
2040 DATA 218,  13,  105, 53,  159, 148, 53,  154
2050 DATA 128,  115, 142, 33,  224, 41,  15,  195
2060 DATA 169,  121, 25,  250, 210, 11,  214, 39
```

Listing 1

In just a moment, I'll describe the bug in more detail; but first, here's how to change Kbd_fix.com to remove this bug.

Repairing the Damage. During a last-minute renumber, I accidentally renumbered a comment statement, making line 1020 a comment instead of a *data* statement. What does this mean? It means that the Basic program, as printed in the November article, reported errors in *data* statements—but reported the wrong line numbers. For example, if you had a data item wrong in line 1370, the Basic program would report bad data in line 1360.

Listing 1 shows the new lines for the Basic program. The comment is now line 1015, between the data lines 1010 and 1020. This listing also contains data for the new, working version of Kbd_fix.com.

To build the corrected Kbd_fix.com, replace the lines of the Basic program as listed in the November article with these lines and run the program once.

Source of the Swarm. Listing 2 shows the changes in the assembly language program. The trouble lay in the old version of INTERCEPT_KEYBOARD_IO. This is a procedure that intercepts calls to the ROM BIOS procedure to read characters from the keyboard (INT 16H, function 0). There are two other functions available under INT 16H (there may be more than two under PC*jr*'s BIOS). The old INTERCEPT_KEYBOARD_IO handled functions 0 and 1 correctly but didn't treat other function numbers properly. To be specific, for any other function numbers, INTERCEPT_KEYBOARD_IO called the ROM BIOS routine with the wrong function number (the function number minus 2) and neglected to restore the BX and DS registers. These bugs are exterminated by the new version of Kbd_fix.

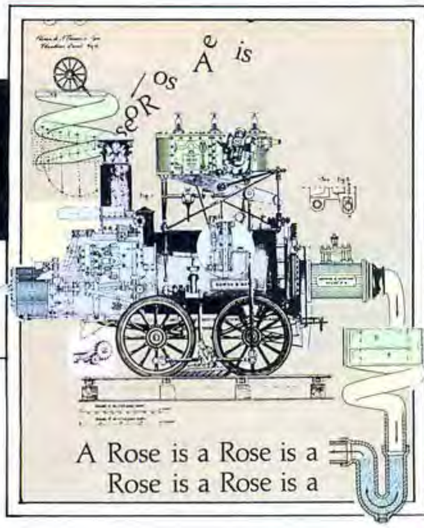
Finally, a word of warning. Kbd_fix.com—even as modified here—doesn't work correctly with Microsoft's *Word*, Lotus Development's *1-2-3*, and *MultiMate*.

Thanks to Earl Robertson for bringing the vermin to my attention, and to two readers on CompuServe who pinpointed the problem: Marty Smith and Jeff Garbers. ▲

```
INTERCEPT_KEYBOARD_IO   PROC   FAR
        STI                                   ; Interrupts back on
        PUSH    DS                            ; Save current DS
        PUSH    BX                            ; Save BX temporarily
        CALL    CHECK_CLEAR_BUFFER            ;Check for buffer cleared
        MOV     BX,CS                         ;Establish pointer to data area
        MOV     DS,BX
        OR      AH,AH                         ;AH=0?
        JZ      READ_CHARACTER                ;Yes, read a character
        CMP     AH,1                          ; AH=1?      **********
        JZ      READ_STATUS                   ;Yes, return the status
        POP     BX                            ;Let ROM BIOS handle other functions
        POP     DS
        ASSUME  DS:NOTHING
        JMP     ROM_KEYBOARD_IO               ;Call ROM BIOS for other functions
                                              ;----- Read the key
        ASSUME  DS:CODE_SEGREAD_CHARACTER: ;ASCII read
        •
        •
        •       READ_STATUS:
        CLI                                   ; Interrupts off
        MOV     BX,BUFFER_HEAD                ; Get head pointer
        CMP     BX,BUFFER_TAIL                ; If equal (ZF = 1) then nothing there
        MOV     AX,[BX]
        STI                                   ; Interrupts back on
        POP     BX                            ; Recover registers
        POP     DS
        RET     2                             ; Throw away flags
INTERCEPT_KEYBOARD_IO   ENDP
```

Listing 2

THE PROCESSED WORD



by Terry Tinsley Datz and F. Lloyd Datz

Palantir

P*alantir*? Maybe you thought it was a video game. From its name, you'd never guess that *Palantir* is a word processor—let alone a word processor that gives you many of *WordStar's* best features while improving on some of its worst. Palantir Software (3400 Montrose Boulevard, Houston, TX 77006; 713-520-8221), formerly Designer Software, has created a program that combines ease of learning and use with many powerful features—proportional spacing, built-in mail merge, and formats that stay with your document, to name a few.

Overall Design. For such a powerful program, *Palantir* has a remarkably simple command structure. You don't need tricolored key stickers or a poster-sized chart to maintain your sanity. A series of simple menus takes you through printing, file handling, and most formatting operations. For things that you do time and again—such as scrolling through your document and reforming paragraphs—you use the function keys. Although some commands require two keystrokes, they begin with F1 or F2 instead of the usual control, alt, or shift. Simply enough, F1 starts or "sets" a new feature, while F2 turns it off or "clears" it. For example, to turn on boldfacing, you press F1-!; to turn it off, you hit F2-!.

Palantir begins with a main menu that serves as the hub of operations. From here you can go to the edit screen, load or save a file, set your document's format, print, run file utilities, use your computer like a typewriter, or get help. Below the menu options are some handy statistics that tell you, among other things, the size of the file you're working on (in kilobytes), the amount of disk space occupied, and the percentage of disk space remaining. To go directly to the editing screen, just press return; to make another choice, you can either move the cursor through the menu or type the first letter of the option. No matter what part of the program you're in, you can retrace your path and return to the main menu by hitting the escape key.

Text Entry and Editing. At the top of the editing screen are a status line and a ruler. The status line displays the cursor's page, line, and column position along with messages that echo the command in progress. At the left of the status line is an arrow that indicates whether the current scrolling and searching direction is forward or backward. Below the status line, a highlighted ruler shows you the left and right margins, the tab settings, and the type of justification in effect.

Palantir gives you good control over cursor movement. As you'd expect, the arrow keys move you by character and by line. One press of the home key takes you to the beginning of the current line and successive presses bounce the cursor between the top and bottom of the screen. To jump to the end of the current line, you hit the right arrow key followed by home.

The return and tab keys move the cursor differently, depending on where you are in your text. When the cursor is within existing text, return takes you to the beginning of the next line without inserting a carriage return and tab takes you to the next word. Otherwise, when there's no text to the right of the cursor, these keys behave as you'd expect.

But it's the *find* command (the F5 key) that fine tunes your control over the cursor. By hitting F5 followed by any character key, you can move to the next occurrence of that character—either forward or backward depending on which way the direction arrow is pointed. Getting to the end of a sentence, for example, is a snap: Just hit F5 followed by a period. To go backward from the cursor position, toggle the direction arrow by tapping F3 (the direction key); then give the *find* command.

A new feature included with our review copy of *Palantir* (and which should be available by the time you read this) allows you to go directly to any page or line number in your document by using the *find* command. This feature is invaluable for getting around in long files. Without it (as in *WordStar*) you must either do repeated scrolling or try to remember a particular word or phrase to search for.

If there is a specific location in your document that you want to return to later, you can define a "home base" for the cursor by pressing F1 followed by home (*set-home*). Then, whenever you want to "home" the cursor, you press F5 and home.

You can do vertical scrolling a number of ways: to the beginning or end of the document and backward and forward by line, by screen, and by page. To help you remember these commands, *Palantir* assigns them to the inside column of function keys, in order of increasing magnitude—F4 scrolls by line, F6 by screen, F8 by page, and F10 to the top or bottom of the file. Just as with the *find* command, the F3 key switches your direction between backward and forward. You can also scroll horizontally—up to 250 columns—to edit extra-wide documents.

Palantir is normally in overtype mode; to insert text, you press the insert key. As you type, existing text is forced ahead rather than split at the point of insertion as with some other word processors.

The delete and backspace keys work as you'd expect: Delete removes the character underneath the cursor and backspace erases the character to the left. To delete to the end of the current line, you hit F2 (*clear*) followed by F4 (the line-scrolling key); *Palantir* automatically reforms your paragraph to close the space. Unfortunately, there's no quick way to delete a single word or an entire line.

To remove more substantial amounts of text, you delete by block. Put the cursor at either the beginning or end of the block and press F1 followed by the delete key (*set-delete*); then move to the other end of the block using the arrow or scrolling keys or the *find* command. Suppose you want to delete to the end of the current page: Make sure that the direction arrow is pointed to the right and tap the F8 key (the page-scrolling key); the cursor jumps to the end of the page, highlighting the text as it goes. If you change your mind, just press escape to unhighlight the block.

Once you've marked the block to be deleted, just press return to remove it. *Palantir* helps you out by reforming your paragraph to

TRIX ARE NOT JUST FOR KIDS

Everyone will be hooked

Kids, watch out for grown-ups

Insoft's TRIX series is educational fun for the whole family. Ten-year-olds and grandparents alike will lock horns with the computer in testing their word skills, their memories, and their quick wits.

Parents, watch schoolwork improve

Kids will pay attention to vocabulary and history lessons after TRIX moves in. They'll eat up new words, facts, and thinking skills for their next bout with TRIX series.

Three TRIX to take on

Wordtrix expands your vocabulary as you work against the clock to form as many words as you can from a grid of random letters. There's only one catch: the computer has a built-in dictionary!

Mystrix is the ultimate mystery-adventure. Given a few clues and a sophisticated crime laboratory, you apprehend and question witnesses, arrest suspects, and use your best deductive reasoning to solve hundreds of baffling crimes.

Quotrix challenges you to beat the computer at figuring out famous quotations. Starting with the author and format, you unravel a trail of clues with the help of crossword puzzles, trivia questions, foreign word translations, and other challenging word games.

Start easy, or be brave

TRIX series offers several different skill levels. You can test the waters as a beginner or try your hand as an expert. How much challenge can you stand?

Play TRIX at your local computer dealer today. Or call us at **(503) 641-5223.** Learning was never so habit-forming.

TRIX Series: Educational Fun From Insoft

insoft

Insoft, Inc.
P.O. Box 608
Beaverton, OR 97075

TRIX series works on IBM PCs and most compatibles.

compensate for the missing text. It doesn't, however, store the deleted material in a buffer, so anything you delete is gone forever.

Moving or copying a block is similar to deleting except that you start by pressing F1-B (*set-block*). There are no limits on block size, and you can even move blocks back and forth between files. After you highlight the block, you're asked whether you're moving or copying. Either way, the program stores a copy of the block in memory. Place the cursor in the new location and press F1-M (*set-move*). *Palantir* inserts the block at the new location and automatically reforms any lines that are affected (another improvement over *WordStar*). For safety's sake, you might want to use this technique for deleting large blocks. Just pretend you're moving the block but don't reinsert it.

Palantir offers just about every search and replace feature you could want. Search phrases can be up to sixty-four characters long and can contain wildcard characters and formatting codes. You have the option of searching for either whole words or an exact series of characters—both forward and backward from the cursor position. When searching for whole words, the program ignores capitalization and surrounding punctuation. In character mode, if your search phrase is "apple", *Palantir* will also find "apples", "apples.", and "crabapple" but not "Apple".

Search phrases can be deleted or replaced once, for a specified number of occurrences, or globally. The program will either ask for permission before each replacement or do the replacements automatically. You also can choose between "quiet" and "noisy" modes. In noisy mode, *Palantir* shows you the replacements on screen; in quiet mode it shows you only the line numbers where replacements are done. Quiet mode is ideal for long documents, since it can halve your search and replace time.

The only noteworthy features missing from *Palantir*'s editor are windowing (split-screen editing capability) and a separate mode for moving columns.

Formatting. With some word processors, you define your document's format by embedding commands; with others, you change the settings on a format menu. *Palantir* combines the two methods: It provides you with menus for formatting large sections of text but uses two-keystroke commands when smaller sections are involved.

To set format values that affect your entire document (such as lines per page or top and bottom margins) you call the formatting menu from *Palantir*'s main menu. These settings can't be varied within the same file.

You can vary other format settings as often as you like by inserting a format line much like the ruler that appears at the top of the editing screen. In fact, if you insert a format line on the first line of text, it becomes your document's ruler. A format line inserted later in your file affects only the text that follows.

To insert a format line or change an existing one, press F1 then F7 (*set-format*) and follow *Palantir*'s prompts to change the left and right margins, set tab stops, select the type of justification, or call a special format menu to make additional changes. *Palantir* offers you the unusual option of right margin *semijustification*—a compromise between ragged right and full justification. Since it inserts only half as many spaces as would be added for full justification, semijustification smoothes out the right margins without letting everyone know that you used a computer.

The special format menu allows you to fine tune your document's layout. You can set line spacing in half-line increments, vary lines per inch and characters per inch, and switch superscripts and subscripts between quarter- and half-line spacing.

You also can vary the number of times the printer strikes each character—from zero times to nine times. To breathe new life into a worn printer ribbon or to produce crisp, camera-ready copy, you can have each character struck several extra times. Or, with the intensity set to 0 (nonprinting mode), you can fill out preprinted forms, one of those odd jobs at which most word processors fail miserably. First, you create a template for your form, leaving blanks where information is to be inserted. Then, to fill out the form, you display it on screen and fill in the blanks, setting off your entries with boldfacing. Since the intensity is set to 0, only the boldface material will be printed on your form.

Special printing features are turned on and off, as you might expect, with F1 and F2. For example, to begin underlining a series of characters, you press F1 followed by an underscore character; to clear the underlining, hit F2 and a second underscore. *Palantir* automatically clears all special printing features at the end of each paragraph, so, if you forget to end one of them, you don't end up ruining your printout. Think of that, *WordStar* fans!

Another handy formatting command allows you to set temporary margins, an invaluable aid for setting off paragraphs or typing outlines or numbered sections. If you go back and insert a temporary margin in previously entered text, *Palantir* will automatically reform the paragraph to fit the new margins. Unlike *WordStar*, *Palantir* keeps track of temporary margin settings even if you move the cursor outside the margins; it also keeps temporary margins intact should you need to go back later to reform the paragraph.

Palantir offers several kinds of tabs. As already mentioned, you set standard tabs on the format line. If you go back and change your tab settings, any columns of text that follow are automatically repositioned. This makes it easy to adjust the spacing of your columns without retyping them.

To align numbers by their decimal point (decimal tabbing), press F1 then control-N. This command has the distinction of being the only one in the program that uses either the control, alt, or shift key. Other tabbing commands will center a line of text between the left and right margins or position a line flush with the right margin.

Palantir stores all format settings on disk along with your document, sparing you the trouble of resetting them each time you read a file in from disk. You can also change the program's default settings; if you routinely use a particular set of values, this feature can be a real timesaver.

Or, better yet, if you work with several different kinds of documents—such as outlines, memos, letters, or reports—you can create any number of permanent files, each of which contains a set of custom format settings. This is similar to copying a block of text between files. Just create a format line for each document type you work with; define each format line as a block of text and give it a filename. Then, whenever you need to use a specific format, copy the appropriate file at the beginning of your document and you're ready to go.

What you see on the screen corresponds to the printed page—with the exception of full and semijustification, line spacing other than single spacing, and some special printing features such as superscripts, subscripts, and proportional spacing. Unlike *WordStar, Palantir* displays both underlining and boldfacing on screen.

Display of page breaks deserves special mention. Until you issue a pagination command, your text is just a series of lines, without page breaks. To paginate, you have to move the cursor to the top of the document and hit F1-P. *Palantir* quickly goes through the file and inserts dotted lines on screen to indicate page breaks.

The extra step required for pagination is well worth your effort, however, since it allows the program to prevent single lines of paragraphs from being isolated either at the top or bottom of the printed page (such lines are usually called *widows* when they appear at the top of the page and *orphans* when they occur at the bottom); only occasionally does *Palantir* let an orphan or widow slip by. You can also insert conditional commands to keep charts and tables from being split between two pages.

Although *Palantir* automatically reforms paragraphs after you change tab stops or do block moves, it doesn't do so during routine editing. To clean up, you have to move the cursor to the first line of a paragraph and press F7. If you need to reform an entire file, just place the cursor at the beginning of the document and then hit F7.

Previous versions of *Palantir* offered little help with hyphen placement other than letting you insert ghost hyphens that printed only if the word fell at the end of a line. The copy of *Palantir* provided for this review, however, included a preview of hyphen-help that will be available with version 1.2. This new feature will allow you to define the size of a hot-zone that consists of the last few spaces at the end of each line. As the program reforms your paragraphs, it stops on each word that extends past the hot-zone and prompts you to hyphenate. You can either accept the hyphen placement suggested by the program or move the hyphen to a different breaking point. By choosing a number between 1 and 9, you indicate how many spaces you want in the hot-zone. A small hot-zone means that more words will be hyphenated, making your right margin more even.

Headers and footers can be of unlimited length, and you can format them any way you want. You can, for example, have them printed book-style on odd- and even-numbered pages. Automatic footnoting, however, isn't supported.

Printing. *Palantir* allows you to print without saving to disk, an option you'll find especially convenient when you're working on a series of letters or memos. You can print partial files or multiple copies (up to 32,000!) of the same file if you wish. *Palantir* is one of the few programs that lets you choose your printer at print time. This feature gives you the flexibility of being able to switch between a dot-matrix and letter-quality printer, for example, without setting up a second program disk. One thing *Palantir* won't let you do is edit one file while printing another (background printing).

Most special printing features are supported, including boldface, double underscore, overstrike, strikethrough, and ribbon shift. You can also print characters that aren't on the keyboard by hitting F1-X and typing the control codes.

Palantir does an excellent job of supporting true proportional spacing. Such support is a programmer's nightmare, since there's no standardization among printers or their printing elements; nevertheless, *Palantir* comes with space tables for a variety of letter-quality printers and their wheels and thimbles—including Diablo, C. Itoh, Qume, and NEC. To use proportional spacing in a document, you

first go to the formatting menu and enter the name of your print element's font (Emperor, Keepsake, Boldps, or whatever), adjust the left and right margin settings, and change the characters-per-inch setting to 0. Then you have to reform and repaginate before you print the file.

File Handling. If you're in the habit of cramming five book chapters into one file, *Palantir* will happily comply. Because it writes sections of your document to disk whenever the available memory is filled, the only limit to your file size is the amount of space on disk. If, on the other hand, you prefer to work with shorter files, you can chain several of them together and print them as one document.

Palantir isn't so generous when it comes to filenames: You're allowed only eight characters. Unfortunately, you can't use the usual three-character extension since that's reserved for the program's own purposes.

Palantir does give you several convenient file-handling features. One of the handiest of these allows you to save your work and continue editing with the cursor repositioned exactly where you left it. Another lets you view the contents of a second file while editing a first. There are also utilities for copying, renaming, and erasing files without returning to DOS. In the event that you try to put too much information on one disk and get the dreaded "Disk Full" message, you can erase as many files as needed or even copy them to another disk without losing the contents of memory. In addition, there's a new-disk command that lets you change floppies in the middle of editing. *WordStar* isn't so obliging.

Another handy feature allows you to insert frequently used words and phrases into your document with only two keystrokes. You can create any number of files (called *lexicon files*) to store such standard sections of text. Each file can include up to thirty-six different expressions (one for each letter and number), and each expression can have as many as 250 characters—enough for whole sentences or fairly short paragraphs.

To create a lexicon file, just go to the editing screen and type in expressions that you use over and over again (your return address, your name and title, and your company's name, for example). You begin each entry with a code number, either a letter or digit; it's best to use mnemonics when possible (*A* for address, perhaps, or *C* for company name). When you want to insert one of these expressions into your document, just hit F9 and enter the appropriate code. If you can't remember what code characters you've used, you can display the lexicon file on screen to refresh your memory.

Mailout. *Mailout* is a series of commands that give *Palantir* powerful file-merging capabilities. These commands, which work much like a mini programming language, allow you to nest files and generate customized form letters and reports. On the simplest level, by typing *include "Filename"* wherever you want another file inserted, you can boilerplate a series of standard paragraphs into a final document or chain several book chapters end to end.

But *Mailout*'s specialty is form letters. You can supply data from the keyboard at print time, or you can have information inserted automatically from a data file. Either way, you create the template for your form letter (or any other standard document) by inserting variable names wherever you want data merged. At the top of the letter, you insert commands that tell *Palantir* the name of the data file you're using and which variables from that file are to be included.

You can create data files with *Palantir*'s editor, using either line format (fields delimited by commas) or Basic format (fields delimited by carriage returns). Data files from other programs will also work, providing you convert the file to a format that *Palantir* can read. The manual gives you specifics for converting your *dBase II* files.

Although *Mailout's* conditional commands are somewhat tricky to learn, with a little patience and imagination, you can put them to work in a number of interesting ways. A simple command such as *if state = "NY" include "Sales.tax"* tells *Palantir* to insert sales tax information for customers who live in New York. A similar, although more complex, command will print different letters for overdue accounts depending on the amount owed and the due date.

Since you can design your own screen notes and include detailed instructions for printing form letters or reports, you can make the program accessible to a totally inexperienced user. Such messages might give the operator general instructions and then prompt for specific information—such as today's date or the name and title of the person signing the letter.

Mailout also allows you to change your document's format and then print it without having to reform manually. You can do this by typing a dummy mailout command at the beginning of your file. *Palantir* will then print your file using the new format values.

Documentation and Support. *Palantir's* documentation includes an eighty-page reference manual, a five-lesson training section, an installation guide for the PC, several sample files on disk, and a seven-page command booklet. Since *Palantir* has versions for several different microcomputers, the command booklet takes a generic approach and lets you fill in the key assignments yourself. You also get a set of key stickers, which—to *Palantir's* credit—you won't need after the first week.

Although the reference manual is written in a friendly, easy-to-read style, it's short on details and examples, especially in the section explaining the *Mailout* commands. The training section, what there is of it, is excellent. Unfortunately, it concentrates on the simpler commands, omitting the lexicon and mailing list features altogether.

Palantir comes with ninety days of toll-free support; you'll find technical representatives readily available and thoroughly knowledgeable. The disk isn't copy-protected.

Ease of Learning and Use. For all its power, *Palantir* is surprisingly easy to learn. The commands are well thought out. For example, what could be more logical than pressing F1 to begin a new feature and F2 to turn it off?

Palantir is also very, very forgiving; there's virtually nothing you can do to bring on disaster. Although there's no provision for automatic file backup, you do get plenty of messages that help you avoid such catastrophes as quitting the program without saving your work or saving a file under an existing filename. You can call help screens from any point in the program. You can even customize the help screens to fit your own applications.

Unlike most other easily learned programs, *Palantir* won't get on your nerves once you've used it awhile. There's nothing clumsy about it, unless you're bothered by having to reverse direction to scroll or search backward.

Updates. Palantir Software has several improvements and two additional programs in the works. Version 1.2 will offer a find line and page feature and hyphen-help, as well as an improved manual with more examples. In addition, you'll be able to run another program from the main menu and define fixed-length records.

Enhancements to look for in the immediate future are a file handler and a spelling checker. The file handler will make *Mailout* easier to use; you'll be able to design the format for data files and sort them based on key information. The speller will be especially welcome, since popular spelling checkers such as *The Word Plus* and *Proofreader* have trouble with some of *Palantir's* hidden codes.

Audience. If you need a powerful word processor—whether you're totally new to word processing or a seasoned pro—*Palantir* is a program to consider. It's an especially worthy candidate for those who need proportional spacing. And, for *WordStar* users who are tempted to switch, *Palantir* provides a conversion program that makes your *WordStar* files readable. The table accompanying this review, incidentally, lists some of the salient advantages of *Palantir* compared to those of *WordStar*.

System Requirements. *Palantir* requires an economical 56K of RAM and, for best performance, at least 400K of disk storage. It works with both the Compaq and Corona, as well as the PC. More than twenty-five printers are directly supported, and several cut-sheet feeders can be used without embedded control codes.

Some Advantages of *Palantir*

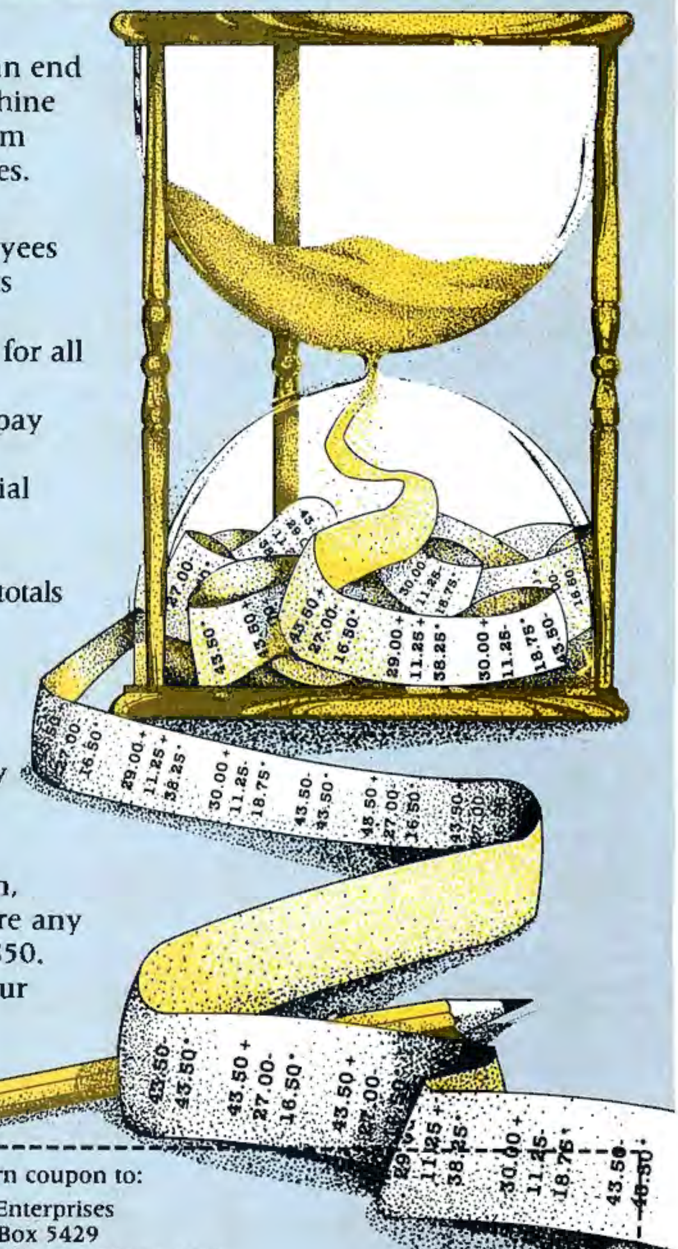
- Proportional spacing
- Format stays with document
- Lexicon files
- Semijustification
- Orphan and widow control
- Multiline headers and footers
- Choose printer from editor
- Print without saving to disk
- Print multiple copies
- Preprinted forms mode
- Display another file while editing
- Rename or copy files and change disks while editing
- Double underscore
- Right flush tabs
- Go to page and line number (version 1.2)
- Set hyphen hot-zone (version 1.2)
- Scroll by page
- Toll-free hot line
- Mailing program at no extra cost

Some Advantages of *WordStar*

- Column block mode
- Background printing
- Redefinable function keys
- Automatic file backup
- Three-character extension for filenames
- Deletion by word and by entire line
- Move cursor backward by word
- Margin release
- Integrated table-of-contents generator and indexer available
- Other programs that can exchange data (*CalcStar*, *DataStar*, etc.)

Palantir (version 1.15) $450
Palantir Software, Inc.
3400 Montrose Boulevard, Suite 718
Houston, Texas 77006
(713) 520-8221 ▲

Is Time one of your Biggest Payroll Problems?



With Payroll Check and your IBM PC you can put an end to hours of pencil pushing and yards of adding machine tape. Payroll Check is the easy-to-use payroll program written especially for small to medium-size businesses.

PAYROLL CHECK ✔

- stores and recalls pay history for up to 500 employees
- prints checks and all other necessary detail reports
- prints W-2's and quarterly reports
- enables you to update federal and state tax tables for all 50 states
- compiles and prints departmental and employee pay summaries for job cost accounting
- comes with a user manual which includes a tutorial section and quick reference section
- has superior menu design
- provides month-to-date, quarter, fiscal and calendar totals
- is IBM XT, Eagle PC, Eagle PC-XL, Compaq, Corona and Hard Disk compatible.

And Best of All — PAYROLL CHECK IS FAST!!

At Key Enterprises we took the time to develop a program that will save you countless hours each pay period. Fully developed and tested, Payroll Check is the next best thing to buying time itself.

So visit your IBM dealer and ask for a demonstration, or order your own Payroll Check demo disk ... before any more of your valuable time slips away. Demo Disk $50. Price applied as credit if you return the disk with your order.

Key Enterprises gives full support to licensed users.

Systems requirements: 128K, two double side disk drives, DOS 2.0, 132 col. printer or 80 col. printers with compressed mode capabilities.

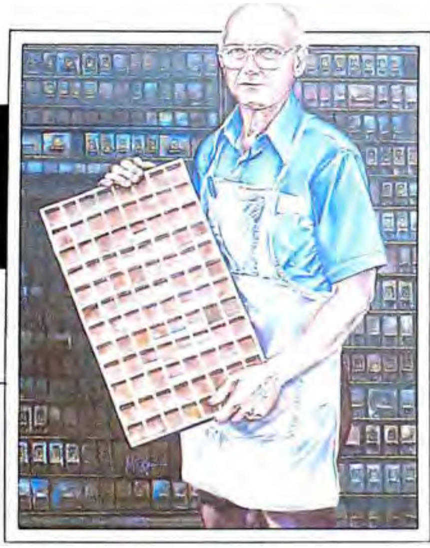
Return coupon to:
Key Enterprises
P.O. Box 5429
Amarillo, Tx. 79117-5429
(806) 372-8962

Name__________ Payroll Check ✔ $495.00

Address __________ Demo Disk $ 50.00

City __________ State _____ Zip _____

Make check payable to Key Enterprises



The Printed Word

by John Dickinson

The Intelligent Printer, Part III

The Intelligent Printer series has been discussing printers in general and how they interface with the PC, and a lot of ground has already been covered. This month we'll start talking about how you can apply your new knowledge to making better (and more intelligent) use of your own printer, and how to develop a handy reference card for it at the same time. We'll start by summarizing what's been discussed so far and formalizing some definitions.

If you've been following the series, you've learned that your printer's built-in computer is intelligent enough to obey a series of commands that are sent to it by your PC, and that these commands are defined as ASCII character codes. To the PC the ASCII character codes to which your printer responds are just ordinary data. Whether you are using the PC-DOS *copy* command or your word processor, every time you print something you are telling the PC to send data to a special file it knows as Lpt1:.

But to your printer's built-in computer, each character code represents a command telling it to print a character or to take an action. The action will either move the paper and printing mechanism around so that the characters form words and numbers, or invoke one of your printer's features to change something about the way printing is done. The correct commands will cause the printer to print the words and numbers that you want printed and print them in the format that you expect.

In the October column you used the PC-DOS *copy* command to turn your PC and printer into a "typewriter" that could do ordinary printing. This was no trouble, since most characters used for typing fall in the printable range of ASCII character codes (032 through 126); they're all on the PC's keyboard and you can just type them in. Using some of the nonprinting action codes such as carriage return (013), line feed (010), and form feed (012) wasn't difficult either. These simple action codes can easily be generated by the PC's keyboard, either automatically by DOS or manually with the help of the PC's alt key and numeric keypad; the codes for these actions, furthermore, are standard in nearly all ASCII printers.

Things got a bit more complicated when you had the *copy* command tell your printer to use an enhanced printing mode, which required that you send your printer a special type of action code called an *escape sequence*. DOS and many PC applications make it difficult for you to send the escape character (ASCII 27) required for escape sequences to the printer, and you had to call on Basic to help. Going by way of Basic was clumsy, but at least it enabled you to send an escape sequence to tell your printer to use an enhanced print option.

Whenever you send ASCII characters to your printer (however difficult that may be), the important thing to remember is that some of the characters you send are interpreted by the printer's computer as action commands and others as data to be acted upon. Normal characters in the printable ASCII range cause the printer to do both. Each of these characters tells the printer (a) to print and (b) to use a particular character (the one you told it to print). A sequence of printable characters will cause a word or number to be printed. Other sequences of characters will cause the printer to take a nonprinting action. You are, in fact, "programming" your printer to do something every time you print a character or sequence of characters.

Just like your PC or any other programmable computer, your printer's computer has its own command (programming) language. The language associated with printable characters is practically universal among makes and models of printers. Also universal are the more common nonprinting commands, such as line feed (LF) and carriage return (CR). Not so universal is the command language for the special nonprinting actions that tell the printer to use its features and options. Much like ordinary computer programming languages, the printer commands and data formats for using their features vary quite a bit from one printer to another.

One thing that does not vary among printers is that programming their features is always done with a series of ASCII characters called a *command sequence*. We can generally define a printer command sequence as a set of one or more ASCII character codes that are not printable characters but have some other meaning to your printer's computer. Escape sequences are the primary (but not only) type of command sequence used by modern ASCII printers. The escape character is always the first character of an escape sequence; it signals the printer to stop printing and pay attention to a nonprinting command. Simple one-character nonprinting codes, such as line feed (010), can also be thought of as command sequences. Some printer models also use one-character codes for commands that are more commonly programmed with escape sequences.

Last month's brief foray into printer graphics explored one of the most sophisticated features of today's dot-matrix printers. The subject may have been a little advanced (or irrelevant) for some readers, but you may have noticed the amount of extra discussion that was necessary to describe how to use the sample graphics programs on printers that are not Epson or IBM compatible. Printer graphics options in different models not only use different command sequences, but also vary in how the graphics data are presented to the printer.

Less sophisticated commands, like graphics, also vary from one printer model to the next. A close look at different printers would show you that command sequences differ quite a bit in their meaning among printers. For example, when you read about using an enhanced print mode in the November edition of this column, a different command sequence was discussed for each of the printers mentioned. The IBM/Epson comparison table printed in the September issue is a more telling example of the differences between printers: These printers are often thought of as identical twins, yet they use different command sequences for some of their features. Many printer features require data,

and the data formats required can also vary quite a bit.

Just as you had to learn the language of PC-DOS and other systems designed for your PC in order to use them, you have to learn your printer's particular language of command sequences in order to use it and its features effectively. Since you've already met the challenge of learning your PC, this may not sound too bad. And it won't be—until you pick up your printer's user manual!

If your manual is typical, you'll face a seemingly hopeless hodgepodge of technical mumbo-jumbo intermixed with highly detailed instructions on how to load paper into the machine, plug it into a wall outlet, and place it on a flat, clean surface. Most of the technical stuff seems to involve DIP switches (whatever they are!), pin signals and voltages, and the correct wiring schematic for your printer's cable.

You're probably able to handle tractor-feed mechanisms and paper pretty well (in most cases it's only slightly more difficult than loading paper into an ordinary typewriter). You hope your dealer supplied you with the right cable and correctly set those mysterious DIP switches for your PC, and you almost certainly have little difficulty with wall plugs and table tops! So, you'll probably skip through all of this without reading it. That's not a bad idea, at least for now.

Somewhere toward the back of the manual (often in a lonely appendix) you will probably (but may not) find an unobtrusive table titled "control codes," "ASCII escape codes," or something equally terse and technical-sounding. It may not look like it, but this table is a gold mine of information about your printer and its features. This is where you can find the command sequences your printer is programmed to understand, and which you have to understand in order to take advantage of your printer's features.

If you're lucky, the codes in the table will be the correct ones for your printer (unfortunately misprints are common in printer manuals). If you are really lucky you will be able to understand them. Let's try to make sure that you're really lucky!

(Many people got really lucky in the manual department when they bought Epson MX-80 Graftrax printers. David Lien's user manual is a delight. Recent purchasers of Epson FX-80 printers are equally lucky because it is now being shipped with the newly published and well-written manual by David Kater. Earlier buyers of this machine might try writing to Epson America to obtain a copy of the newer manual.)

There are usually two things that make command sequences tables difficult to understand. The first is the way the tables are organized. The other is the various, and often confusing, data formats in which command sequences are presented. Both of these problems are also found in the text of most manuals.

Let's look at the organizational problem first.

For some reason it has become traditional for authors of printer manuals to organize the command sequences (that's what we'll call them here, regardless of the name used in your manual) into a table ordered according to the ASCII sequence of the character codes used for the commands. The table is usually split into two parts: one-character command sequences (those that don't require the escape character) and escape sequences.

The first part of the table usually looks something like the standard table we printed in October. It contains the standard nonprinting commands found in most PC-compatible printers, possibly along with some other one-character commands, with the commands ordered in their ASCII character sequence. The second part of the table contains escape sequences, and is ordered in the ASCII sequence of the character codes that follow the escape character in the command sequences.

This type of organization can be useful for some types of reference work or indexing of the printer control codes. However, for use in understanding your printer, or even in finding a command sequence when you need it, it's a poor system. The reason is that printer designers usually have assigned the ASCII character codes for command sequences to features on a first-come, first-served basis. As a new feature is implemented in a printer model, it is often assigned the first available code in the ASCII sequence.

As a result of this ad hoc design strategy, the functional relationships that exist among a printer's features are difficult to discern from these tables. Features affecting one functional area, such as vertical spacing, are mixed in with features affecting a different function, such as character print enhancement.

The first step in gaining a better understanding of your printer's command language and in developing your printer reference card is to reorganize the table of command sequences. The most useful way to organize the table is to group the command sequences into their respective functional areas. When this reorganization is complete, the functional relationships among the features will be clear and your printer will be easier for you to understand. You'll be able quickly and easily to find the command sequences required for the features you want to use.

For an example we'll reach back into history and use the most popular printer ever sold in the microcomputer industry, the Epson MX-80. Although Epsons are no longer sold without the Graftrax feature, we'll use the older model as an example, because it's simpler than the Graftrax-equipped models and it contains enough features to make the example useful and interesting. Besides, many of you probably have these earlier Epsons hooked up to your PCs! (We'll exclude all MX-80 features that have no application for PC users.)

You can either go through this example and follow the exercise in detail or try to follow it by using your own printer's manual. It might be a good idea to do both, since you can easily get mired in something you don't understand in your printer's manual and the example used here may help you to figure the problem out.

To start, table 1 is a slightly modified reproduction of the table of command sequences presented in Appendix I of the MX-80 user's manual (the command sequences are called function codes there). The modifications were made to clarify the descriptions of the commands, but table 1 is printed in the same order as in the manual. You'll notice that table 1 is exactly as we described—the one-character command sequences that are fairly common among printers, along with some one-character commands specific to the MX-80, are listed first, followed by

One-Character Commands	BEL	Sound Printer Horn
	HT	One Horizontal Tab
	LF	One Line Feed
	VT	One Vertical Tab
	FF	Form Feed
	CR	Carriage Return
	SO	Enable Wide Characters
	SI	Enable Compressed Characters
	DC2	Disable Compressed Characters
	DC4	Disable Wide Characters
	CAN	Clear Printer Buffer
Escape Sequences	Esc 0	Set Spaces at 1/8"
	Esc 1	Set Spaces at 7/72"
	Esc 2	Set Line Feed at 6 LPI
	Esc 8	Disable Paper-Out Switch
	Esc 9	Enable Paper-Out Switch
	Esc A	Set Line Feed "N"/72"
	Esc B	Set Vertical Tabs at "N1,N2,,,NK"
	Esc C	Set Forms to "N" Lines
	Esc D	Set Horizontal Tabs at "N1,N2,,,NK"
	Esc E	Enable Emphasized Characters
	Esc F	Disable Emphasized Characters
	Esc G	Enable Double-Strike Characters
	Esc H	Disable Double-Strike Characters

Table 1. Epson MX-80 Printer Command Sequences
ASCII Character Code Order

the escape sequences used for most of the more advanced features found on the MX-80.

The order of the command sequences in both parts of the table is the ASCII character order (check Appendix G of your Basic manual if you want to be sure of this). The names printed to the left are the ones used in the manual; they reflect the characters that are used in the command sequences. If the names in the table make no sense to you, don't worry. In a moment we'll be giving the MX-80 command sequences (and the ones for your printer, if you're doing the exercise for it) new names.

If you scan through the table, you'll see a few groups of command sequences that do seem functionally to belong with each other. For example, all of the command sequences used by the MX-80 for enhanced printing modes (escape-E, escape-F, escape-G, and escape-H) are at the bottom of the table. This is purely by chance (they were probably developed at the same time), and, overall, the table is something of a jumble; the command sequences are in no special functional order.

To make better sense of the table, search through it for all MX-80 command sequences related to character printing. You should find all commands that either change the size (font) of the characters or enhance their printed darkness (these commands should be clear from the function descriptions). Then put these commands into a separate list. Table 2 illustrates this exercise, moving the MX-80's eight commands to

BEL	Sound Printer Horn
HT	One Horizontal Tab
LF	One Line Feed
VT	One Vertical Tab
FF	Form Feed
CR	Carriage Return
SO	Enable Wide Characters >
SI	Enable Compressed Characters >
DC2	Disable Compressed Characters >
DC4	Disable Wide Characters >
CAN	Clear Printer Buffer
Esc Ø	Set Spaces at 1/8"
Esc 1	Set Spaces at 7/72"
Esc 2	Set Line Feed at 6 LPI
Esc 8	Disable Paper-Out Switch
Esc 9	Enable Paper-Out Switch
Esc A	Set Line Feed "N"/72"
Esc B	Set Vertical Tabs at "N1,N2,,,NK"
Esc C	Set Forms to "N" Lines
Esc D	Set Horizontal Tabs at "N1,N2,,,NK"
Esc E	Enable Emphasized Characters >
Esc F	Disable Emphasized Characters >
Esc G	Enable Double-Strike Characters >
Esc H	Disable Double-Strike Characters >

Character Font and Print Control
Functional Area

> SI	Enable Compressed Characters
> DC2	Disable Compressed Characters
> SO	Enable Wide Characters
> DC4	Disable Wide Characters
> Esc E	Enable Emphasized Characters
> Esc F	Disable Emphasized Characters
> Esc G	Enable Double-Strike Characters
> Esc H	Disable Double-Strike Characters

Table 2. Epson MX-8Ø Printer Command Sequences
Moving Commands to Their Functional Area

Compute and print client tax returns in minutes

on your microcomputer with
*MICRO-TAX**

That's right, in just minutes you can have a client's completed tax return in your hand.
Think about it... you increase client volume, you increase your profits. Plus, you save the cost of your computer service bureau—and you have complete client security.

FEDERAL AND STATE PACKAGES TO MEET EVERY NEED. MICRO-TAX* offers four Federal tax packages and 25 state packages (fully integrated with the Level II Program), so you can select the programs that best meet your needs:
Level I—Federal Individual Package: for individuals preparing their own taxes.
Level II—Federal Professional Individual Package: for accountants, registered agents, tax attorneys, and other tax professionals.
Level III—Federal Partnership/Corporate Package: for those who prepare Federal Partnership, Corporate, and Subchapter S returns.
Level IV—Overseas Tax Package: addresses the unique tax situations of United States Expatriates.
Levels II, III, and IV have a depreciation module and automatically compute underpayment penalties and minimum tax. In addition, Levels II and III automatically compute self-employment taxes, and Level II computes income averaging.

FLEXIBLE DATA ENTRY. With MICRO-TAX* you can organize data entry in a sequence similar to that of manual tax preparation, or you can choose another sequence. The menu driven system makes data entry simple.

MULTIPLE PRINTING OPTIONS. You can input client tax information at the time of interview and produce forms immediately, or enter data during the day and batch print returns at night. MICRO-TAX* prints your returns on IRS forms, IRS approved substitute forms, or with transparent overlays.

TAXNET*—TELETEXT SUPPORT NETWORK. MICRO-TAX* customers can now have access to an electronic mailbox and instantaneous memoboard through the TAXNET* teletext support network. With TAXNET,* you can send information, ask questions, get answers and updates—directly through your computer and a modem.

TAX ORGANIZER. Now MICRO-TAX* offers a Tax Organizer. You get both the software and the forms, so each year, you can send your clients an organizer with the prior year's client data printed on it.

HARDWARE COMPATIBILITY. MICRO-TAX* is compatible with your IBM PC/XT,* DEC Rainbow,* Radio Shack,* or any other personal computer with CP/M-80,* PC DOS,* or MS DOS*—from Apple* to Zenith.*

So, take the tedium out of tax preparation—save time and money—*Call Micro-Tax* for complete details, or call your local dealer.*

FULL FEDERAL MICRO-TAX* PERSONAL COMPUTER SYSTEMS	1983 FEDERAL FORMS AND SCHEDULES INCLUDED																															PRINTS			
	1040, 1040A, 1040ES	1065, 1120, 1120S	A, B, C, E, G, SE, W	D, F	K, K-1	R, RP	1116	2106	2119	2210	2220	2440	2441	2555	3468	3903	3903F	4137	4255	4562	4626	4684	4797	4835	4952	4972	5695	6251	6252	90-22.1	ACRS	BATCH	Substitute Forms	IRS Forms	Blank Paper	Continuous Preprinted Forms
Level I, Individual $195.	•		•	•				•					•														•						•	•	•	
Level II, Professional Individual $1000.	•		•	•		•	•	•	•	•		•	•		•	•		•	•	•		•	•	•	•	•	•	•	•		•	•	•	•	•	•
Level III, Partnership/ Corporate $1000.		•		•	•						•				•				•	•	•	•	•	•					•		•	•	•	•	•	•
Level IV, Overseas $2000.	•		•	•			•	•		•				•		•	•						•	•				•		•	•	•	•	•	•	•

• All forms and schedules subject to final IRS changes for 1983 tax year.
• 25 Integrated state returns available • Orders taken for yearly update packages
• Transparencies: Level I, $150; Levels II, III, IV, $250 each.

*C/PM—trademark of Digital Research, Inc.; DEC Rainbow—trademark of Digital Equipment Corp.; MICRO-TAX and TAXNET—trademarks of Microcomputer Taxsystems, Inc.; MS DOS—trademark of Microsoft Corp.; PC DOS, IBM PC, and IBM XT—trademarks of IBM; Apple—trademark of Apple Computers; Zenith—trademark of Health Company and Zenith Radio Corp.; Radio Shack—trademark of Tandy Corp.

MICRO-TAX* MICROCOMPUTER TAXSYSTEMS, INC.
6203 Variel Avenue, Suite A Woodland Hills, CA 91367 Dept. 11 Phone (213) 704-7800
(Area code changes to (818), effective January 1984)

control character printing into a group by themselves. (Notice that only half of them are escape sequences, while the others are one-character codes.)

There are only two command sequences in this group whose names make any sense to humans (they all make sense to the MX-80). The SI and SO codes, which are inherited from equivalent uses in ASCII communication protocols for time-sharing terminals, mean *shift in* and *shift out* the character font width respectively. But even these names aren't particularly appealing, so let's give the character font and print control command sequences for the MX-80 some names that we can all understand.

Table 3 shows the same group of commands but also includes some suggestions for more meaningful names, names that will be easier for you to remember. These are examples, meant to serve as mnemonics for the MX-80 features controlled by the command sequences. You can give the command sequences for your printer any name you want, but you should try to make the names meaningful so that they can be easily remembered. In the table, *double* means enable double-strike printing, while *wide* means enable wide character printing.

One naming convention used in table 3 reminds you that one command sequence reverses the effects of another. For example, *comp* enables compressed characters, while *xcomp* disables compressed characters. Another naming convention used in table 3 is that names for printer command sequences are limited to eight characters or fewer. This saves space in the reorganized table for inserting more information as we continue the exercise, and it will serve other purposes that you'll learn about in later installments of this series. Keep the old names from the user manual around for the moment. The ones used in the MX-80 manual have some information you'll need later on (although the equivalent names in your printer's manual may not).

Now that you've extracted all the command sequences related to the MX-80's character printing, start extracting the commands for other

SI	COMP	Enable Compressed Characters
DC2	XCOMP	Disable Compressed Characters
SO	WIDE	Enable Wide Characters
DC4	XWIDE	Disable Wide Characters
Esc E	EMPHIZE	Enable Emphasized Characters
Esc F	XEMPHIZE	Disable Emphasized Characters
Esc G	DOUBLE	Enable Double-Strike Characters
Esc H	XDOUBLE	Disable Double-Strike Characters

Table 3. Epson MX-80 Printer Command Sequences Character Font and Print Control Functional Area

functional areas. The commands normally defined for the MX-80 include:

General Printer Control—Commands that affect the general operation of the printer.

Forms (Page) Control—Commands that set form length or eject a form.

Vertical Line Feed Control—Commands having to do with control of vertical paper movement, including vertical spacing and tabulation.

Horizontal Head Control—Commands that move the printing head horizontally or control how far it moves.

Repeat the exercise of putting the command sequences for each functional area into a separate list. When you're done, make up more mnemonic names that describe the function, including reversing effects, but limit the names to eight characters. When all twenty-four MX-80 commands are put in their proper functional areas and renamed, the table should look like table 4.

If you have done the exercise for your own printer, your table may be longer or shorter than table 4, depending on the number of features

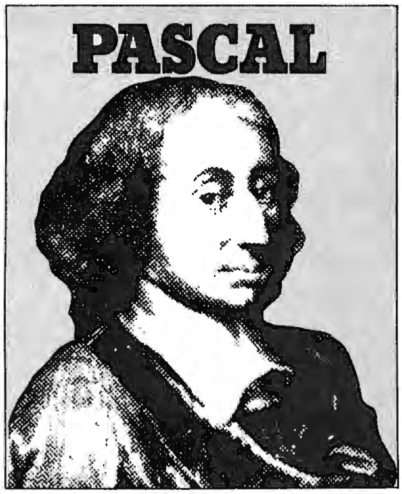
		General Printer Control
BEL	HORN	Sound Printer Horn
CAN	CLEARBF	Clear Printer Buffer
Esc 8	XFORMOUT	Disable Paper-Out Switch
Esc 9	FORMOUT	Enable Paper-Out Switch
		Forms Control
FF	FORMFEED	Form Feed
Esc C	FORMLINE	Set Forms to "N" Lines
		Vertical Line Feed Control
LF	LINEFEED	One Line Feed
VT	VT	One Vertical Tab
Esc B	SETVT	Set Vertical Tabs at "N1,N2,,,etc."
Esc 2	SPC6LPI	Set Line Feed at 6 LPI
Esc 0	SPC8LPI	Set Spaces at 1/8" (8 LPI)
Esc 1	SPC10LPI	Set Spaces at 7/72" (10 LPI)
Esc A	SPCN72	Set Line Feed "N"/72"
		Horizontal Head Control
CR	CR	Carriage Return
HT	HT	One Horizontal Tab
Esc D	SETHT	Set Horizontal Tabs at "N1,N2,,,etc."
		Character Font & Print Control
SI	COMP	Enable Compressed Characters (16.5 CPI)
DC2	XCOMP	Disable Compressed Characters
SO	WIDE	Enable Wide Characters (5 CPI)
DC4	XWIDE	Disable Wide Characters
Esc E	EMPHIZE	Enable Emphasized Characters
Esc F	XEMPHIZE	Disable Emphasized Characters
Esc G	DOUBLE	Enable Double-Strike Characters
Esc H	XDOUBLE	Disable Double-Strike Characters

Table 4. Epson MX-80 Printer Command Sequences Functional Area Order

your printer has. You should try to include all features that are documented in your manual, even if you don't think you're interested in using them or don't understand them. Learning about them won't hurt and may start you thinking about using them in future printer applications.

By now you have the beginnings of a useful printer reference card. The printer's command sequences are organized into areas of related function, and they have names that make some kind of sense to you. There's more to do, however, before your card is complete and your printer's features become more useful to you.

Next month we'll continue decoding your manual by translating the actual command sequence data into something that you can use in each of your printer applications. They are documented in several different ways and used in about an equal variety of ways by application packages and programming languages. You'll be able to add the data formats you need to your reorganized table of command sequences, and by the time you're done you'll have a useful reference card for your printer. ▲



FROM BEGIN TO END

by Bruce Webster and Deirdre Wendt

Two months ago we started a series of columns on file handling in Pascal, looking at aspects specific to IBM and UCSD Pascal. We'll finish up this month with a look at Pascal/MT+ from Digital Research (Pacific Grove, CA). We'll cover five major topics: disk file I/O, random access I/O, untyped files, device I/O, and error handling.

A word before going on: Some material from the last two columns is repeated here; please bear with us. As we said two months ago, we tried to fit all this into one column, but it was just too much. Splitting it into three columns requires recovering the same ground each month. Things will change next month. Honest.

Disk Files. Pascal/MT+ runs under CP/M-86 and MS-DOS (a version is available for each), so it follows the file-naming conventions of these systems: a filename may have as many as eight letters and/or digits and may include an extension of up to three characters. If the extension exists, it is separated from the filename by a period.

The following filenames are acceptable:

```
thisfile
thx1138
simple.pas
simple.exe
stars.dat
```

There are two ways of opening a file under Pascal/MT+. The first is similar to the procedure used in IBM Pascal. You first assign the file variable a filename, using the statement:

```
assign(⟨file⟩,'⟨filename⟩');
```

followed by a call to *reset* or *rewrite*. If the file already exists, you can switch to the statement

```
open(⟨file⟩,'⟨filename⟩',⟨result⟩);
```

which replaces the calls to *assign* and *reset*, and returns an I/O error code to boot (more on these below).

Having gotten this far, you have a few options. First, you can use the statement

```
purge(⟨file⟩);
```

to delete the disk file assigned to *⟨file⟩*. If you want to save it, you can simply close the file with

```
close(⟨file⟩,⟨result⟩);
```

where, again, *⟨result⟩* returns an error code value. If you wish the file to go away when you're done, you can instead use

```
closedel(⟨file⟩,⟨result⟩);
```

which will, as you've guessed, delete the file and return the result code.

Pascal/MT+ File Handling

For the last two months, we've been using a sample program that opened a text file, read in numeric data, opened a binary file, and wrote the data out to it, then closed both files, deleting the text file. Here's that same example rewritten for Pascal/MT+:

```
CONST
    infilename      = 'STARS.TXT';
    outfilename     = 'STARS.DAT';
TYPE
    stars           = ARRAY[1..3] OF real;
    { [1] = x, [2] = y, [3] = z }
VAR
    infile          : text;
    outfile         : FILE OF stars;
    x,y,z           : real;
BEGIN
    open(infile,infilename,result);        { open to read }
    assign(outfile,outfilename);           { open to write }
    rewrite(outfile);
    WHILE NOT eof(infile) DO BEGIN
        readln(x,y,z);                     { read in values }
        outfile∧[1] := x;                  { copy to file variable }
        outfile∧[2] := y;
        outfile∧[3] := z;
        put(outfile)                       { write out to disk }
    END;
    purge(infile);                         { remove text file }
    close(outfile)                         { close and save data file }
END. { of PROGRAM convert }
```

Random Access I/O. You'll often want to read and write data in a file in a nonsequential or *random* manner. For example, having created the file Stars.dat, you might want to be able to retrieve the coordinates of a given star without reading the entire list of stars into memory or resetting the file and reading through all the preceding stars to get to the one you want. The ability to read (or write) directly any record in a file regardless of its location is called *random access*.

Pascal/MT+ allows you to access a file randomly by opening it with *reset* or *open*, then using the commands *seekread* or *seekwrite* to read or write a specific record in the file. For example, say you wanted to open a "star data" file so that you could read the coordinates for any given star. The program might look like this:

```
PROGRAM look_at_stars;
TYPE
    stars = ARRAY[1..3] OF real;
VAR
    datafile              : FILE OF stars;
```

```
      x,y,z               : real;
      starindexi,result   : integer;
   BEGIN
      open(datafile,'STARS.DAT',result);
      REPEAT
      write('Enter star # (0 to exit): ');
      readln(starindex);
      IF starindex > 0 THEN BEGIN
        seekread(datafile,starindex+1);   { move to the star }
        write('Coordinates of star #',starindex:3,': ');
        write('(',datafile^[1]:12:4,',',datafile^[2]:12:4,
              ',',datafile^[3]:12:4,')')
        END
      UNTIL starindex <= 0;
      close(datafile)
   END.
```

The *seekread* statement moves to a specific record in a file and reads it into the file variable, eliminating the need for a *get* statement. The first record is record number 0; that is, *seekread(⟨file⟩,0)* would point at the first record in ⟨*file*⟩. By the same token, *seekwrite* moves to the indicated record and writes out the contents of the file variable, replacing the *put* statement. This means that you must be sure that the file variable has already been assigned its proper value before you call *seekwrite*.

Untyped Files. Up until now, all files we've looked at have been text files (= *FILE OF char*) or data files (= *FILE OF ⟨type⟩*). However, there are times when we want to deal with "raw" data—data that hasn't been formatted for us. For example, in our computer game *SunDog*, we needed to have many different types of data out on the disk. But we didn't want the overhead of having many files open at once or having to open and close files constantly.

The solution? We put everything into one large disk file and used an *untyped file* to access it.

An untyped file is declared as follows:

```
   VAR
      bigfile       : FILE;
```

It's opened and closed just like any other file. Reading from and writing to it are different, though. Untyped files can only be read or written a block (128 bytes) at a time. You do this by means of two procedures, *blockread* and *blockwrite*. Here's their format:

```
      blockread(<file>,<buf>,<result>,<numbytes>,<offset>);
```

We only listed one, because the other has exactly the same parameters. ⟨*file*⟩ is, of course, the untyped file variable (such as *bigfile* above). ⟨*buf*⟩ can be any type of variable; most often, it's an array of some sort. ⟨*result*⟩ returns the I/O operation result code. ⟨*numbytes*⟩ is an integer value giving the number of bytes you wish to read (or write). Since it must be a multiple of 128, it's just another way of specifying the number of blocks you want to read in. For example, if you want to read in three blocks, ⟨*numbytes*⟩ must equal 128 * 3, or 384. You should be absolutely sure that *sizeof(⟨buf⟩)* is greater than or equal to ⟨*numbytes*⟩; otherwise, bizarre things can happen to your program. The last parameter, ⟨*offset*⟩,can be used in two ways. If ⟨*offset*⟩ is set to −1, then all your reads and writes will proceed sequentially through the file. In other words, you'll start at block 0 and move straight through the file. If ⟨*offset*⟩ is greater than or equal to 0, then you'll read (or write) that particular block within the file, allowing you to do random access within an untyped file.

Let's illustrate all this with an example. Suppose that blocks 10 through 13 of your data file contain records which, for convenience' sake, are each thirty-two bytes long. Each block then has four of these records in it, and there are sixteen (four times four) records in all. You could then write the following routine to get or put a specific record (numbered 0 through 15) from that chunk of the file. We'll call the record type *goodrec* and will assume that your untyped file (*bigfile*) is already open.

```
        PROCEDURE goodrec_IO(indx  :  integer;
                             read  :  boolean;
                          VAR rec  :  goodrec);
{
     does read/write for record[indx], where indx is the range
0. .15 and the records are found in blocks 10. .13 of bigfile
     if read = true, then reads record, else writes it
}
     VAR
        bcnt,iblk,irec,ierr    : integer;
        data                   : PACKED ARRAY[0..3] OF goodrec;
     BEGIN
        indx := abs(indx) MOD 16;   { force to allowable range  }
        iblk  := indx DIV 4 + 10;   { calculate block #         }
        irec  := indx MOD 4;        { calculate rec w/in block  }
        blockread(bigfile,data,ierr,128,iblk); { get the data   }
        IF read
          THEN rec := data[irec]    { get appropriate record    }
        ELSE BEGIN
          data[irec] := rec;        { else save it in 'data'    }
          blockwrite(bigfile,data,ierr,128,iblk) { & write it   }
        END
     END; { of PROC goodrec_IO }
```

By writing similar routines for the other data types stored in *bigfile*, you can have ready access (and random access, at that) to a wide variety of data types with a minimum of overhead. If the size of the buffer

(128 bytes or some multiple thereof) bothers you, you can always use *mark, new,* and *release* to create it on the heap and then get rid of it.

Device I/O. Most applications require I/O involving the computer hardware itself. Reading from the keyboard and writing out to the screen are the two most obvious examples. Standard Pascal (and Pascal/MT+) predefines the text files *input* and *output* for just those functions. All *read* and *write* statements without a file variable use these two files. But there are other times when we might want to read from or write to a specific device. How do we do this?

Simple. We use a set of special filenames that refer to hardware devices rather than to disk files. For example, suppose we wanted to modify *convert* so that we could enter the data manually instead of having it read from a disk file. We would simply change the string constant *infilename* to read *CON:*. We'd then enter the data, line by line. When finished, we'd type control-Z, which tells the program that it's reached the "end of file."

By the same token, a program that writes a text file to disk could be redirected to write the output to the screen. Note well: Because these are character-oriented devices, only files of type *text* or *interactive* should be connected with them.

Here's a list of the special filenames that Pascal/MT+ recognizes:

CON:	console I/O, i.e., read from the keyboard and write to the screen. Echoes input, allows correction with backspace. Expands tabs on output. Echoes CR as CR/LF for both input and output.
KBD:	keyboard input with no echo or interpretation.
TRM:	console output with no interpretation.
LST:	the line printer. Can only be used for output. No interpretation.
RDR:	input device, usually an RS-232 port.
PUN:	output device, usually an RS-232 port.

Unfortunately, the Pascal/MT+ documentation doesn't give much more information than we have here. We've played around a little with these, but haven't figured out much more than this.

Error Handling. Pascal/MT+ won't blow up or crash your program when it gets an I/O error. Instead, you can use the function *ioresult* to test for an error and act appropriately. Some procedures, such as *open, blockread,* and *blockwrite,*return the error code as one of their parameters.

The biggest problem with using this information is that the Pascal/MT+ documentation doesn't list what the codes are or what they mean. The only references we could track down said that a value of 255 indicated that the file desired didn't exist. So we decided to play around a little to see what we could come up with. Our results? Well, the only error code we could get it to generate for file access commands was 255. For integer number input, bad data would either be ignored (meaning it was still waiting for the real thing), or it would come back with its own prompt, "Bad integer—try again." Same thing for real numbers, with the appropriate prompt. Frankly, we didn't like the compiler preempting our ability to trap and handle errors; that's something we'd much rather take care of ourselves. Why? Well, if our application involves direct cursor addressing, a message like that would mess up the display. We think that a better solution is to return an error code and let the programmer decide what action to take.

Conclusion. That ends our discussion of files and I/O, at least for a while. Next topic: strings and string handling. Niklaus Wirth did not define a standard string data type in Pascal, so almost every implementation has its own. We'll look at how they're defined and how to use them. See you then. ▲


THE BASIC ASSEMBLY LINE
JTP

GRABBING THE DIRECTORY FROM BASIC

BY HOWARD GLOSSER

It's hard to believe, but Basic doesn't have any command that allows a program to retrieve filenames from a disk directory and store them in an array. The *files* command displays the names of the files residing in a disk directory, but just try to *do* anything with those names. Clever programmers have devised ways to retrieve the directory indirectly—by using the *files* command, reading the screen buffer character by character, and filling an array with the results. But there's got to be an easier way.

Direct Access to the Directory. This month's subroutine, called Directry, allows any Basic program to retrieve a disk's directory. Basic reads all the directory entries and stores them in an array, which can then be listed, sorted, or used in whatever other way you see fit. Directry presents one limitation for those using DOS 2.0; it reads only the root directory of a disk. So if you have disks with subdirectories, you'll have to include a *chdir* command in your Basic program to change directories just prior to the subroutine call for Directry.

Directry makes use of DOS function calls hex 11 and hex 12, which respectively perform an initial and subsequent search of the directory based on a search name located in the file control block (FCB). The search name supplied in this case consists of all question marks (the global search character). By using this type of name, you can retrieve every active entry in the directory (unused and erased entries will not appear).

Creating the Directry Subroutine. The Basic program listed in figure 1 creates the Directry subroutine. It builds the subroutine by poking the assembly language code located in the data statements (lines 450 through 640), one byte at a time, into the string variable Subrt$. To verify that none of the data statements is in error, a checksum by *line is* performed as the data are being read. If an error is encountered, the bad data line is identified in an error message and the program stops. Otherwise, the *bsave* in line 370 writes the subroutine to disk under the name "Directry". The subroutine is then available for use in any Basic program you write.

How To Read a Directory. The Basic program in figure 2 offers an illustration of how Directry might be used. It starts off by defining Subrt$, the string variable where the subroutine will be stored. Next, the string descriptor of Subrt$ is retrieved, followed by the actual location of the string. Line 100 uses the *bload* command to place the subroutine stored on disk into the string variable Subrt$.

Lines 140 through 180 define variables used in the subroutine and the demonstration program. Dirlst$, defined as a one-dimensional array consisting of 112 elements, is where the entries from the directory will be stored as they are retrieved in the Directry subroutine.

Next is string FCB$, where the subroutine will build a file control block. String DTA$, a work area required by the subroutine, follows in line 160. It's used as the disk-transfer area where information retrieved by the hex 11 and 12 function calls is stored while awaiting transfer to the proper Dirlst$ element.

Finally, there is the twelve-byte string Filler$. This is used in line 220 to set each element of Dirlst$ to a length of twelve bytes and a value of blanks. This is necessary because the subroutine expects each element of Dirlst$ to meet these requirements. Also defined is integer variable Count%, which tracks the number of entries retrieved from the directory.

When the actual demonstration program runs, a prompt appears on the screen asking you for the drive letter of the directory you want retrieved. The program can access as many as four drives. If you select drive D: when you have no such drive on your system, under DOS 1.1 the last active drive will be read, and under DOS 2.0 the program will return a directory count of zero entries. If you key in an invalid drive letter (not A through D), the speaker will buzz, and the system will wait for you to give it a valid drive letter.

Once you've selected a drive, lines 410 and 420 find the location of the Directry subroutine and perform the call, passing the necessary parameters to the subroutine. While the program executes the subroutine, the disk spins (if you're accessing a RAM disk, the program pauses slightly) as the subroutine retrieves and stores the directory information in the Dirlst$ array. After the call, line 430 returns control to the main program; a count of the directory entries, along with the directory listing for the drive requested, is displayed on the screen.

If you want to run the program again, specifying a different drive letter, press the space bar. The demonstration stops when you press *S*.

The screen display of files located on the directory is patterned after the DOS 2.0 display. However, that's where the similarities end. The Directry subroutine not only reads the directory read but also stores the directory entries in the Dirlst$ array. This means that your filenames can be sorted and then displayed, used for search or comparison purposes, or accumulated from several disks to provide a comprehensive disk index. Try doing that with the *files* command.

```
10 '      ***** BUILD DIRECTRY *****
20 '
30 '      RETRIEVE A DISKETTE'S DIRECTORY
40 '         FROM WITHIN A BASIC PROGRAM
50 '
60 '      WRITTEN BY HOWARD GLOSSER
70 '
80 CLS
90 PRINT "Creating DIRECTRY subroutine. . .":PRINT
100 '
110 '   ** THIS SETS UP STRING LOCATION FOR SUBROUTINE
120 '
130 DEF SEG
140 SUBRT$ = STRING$(160,32)
150 SUBLC% = VARPTR(SUBRT$)
160 DRCT = PEEK(SUBLC% + 1) + PEEK(SUBLC% + 2) * 256
170 LCN = DRCT
180 '
190 '   ** THIS BUILDS THE SUBROUTINE
200 '
210 LINENO% = 450
220 FOR STMT = 1 TO 20
230    FOR MEM = 1 TO 8
240       READ DT%
250       POKE LCN,DT%
260       CHECKSUM% = CHECKSUM% + DT%
270       LCN = LCN + 1
280    NEXT
290    READ DT%
300    IF CHECKSUM% <> DT% THEN 410
310    LINENO%=LINENO% + 10
320    CHECKSUM% = 0
330 NEXT
340 '
350 '   ** THIS SAVES THE SUBROUTINE
360 '
370 BSAVE "DIRECTRY",DRCT,&H9F
380 PRINT "DIRECTRY SUBROUTINE CREATED"
390 END
400 '
410 PRINT "ERROR in DATA STATEMENT - Check line " LINENO%:END
420 '
430 '   ** DATA STATEMENTS TO BUILD SUBROUTINE
440 '
450 DATA &HEB,&H07,&H90,&H00,&H00,&H00,&H00,&H00,&H0182
460 DATA &H00,&H55,&H8B,&HEC,&H8B,&H5E,&H0E,&H8B,&H034E
470 DATA &H77,&H01,&H8B,&H04,&H35,&H40,&H00,&H8B,&H0207
480 DATA &H5E,&H0C,&H8B,&H7F,&H01,&H88,&H05,&H57,&H0259
490 DATA &H47,&HB0,&H3F,&HB9,&H0B,&H00,&HF3,&HAA,&H0397
500 DATA &H2E,&HC7,&H06,&H05,&H01,&H00,&H00,&H8B,&H018C
510 DATA &H5E,&H0A,&H8B,&H57,&H01,&H2E,&H89,&H16,&H0218
520 DATA &H03,&H01,&HB4,&H1A,&HCD,&H21,&H8B,&H5E,&H02A9
530 DATA &H08,&H2E,&H89,&H1E,&H07,&H01,&H2E,&HFF,&H0212
540 DATA &H06,&H07,&H01,&H5A,&HB4,&H11,&HCD,&H21,&H021B
550 DATA &H3C,&HFF,&H74,&H1A,&H2E,&HFF,&H06,&H05,&H0301
560 DATA &H01,&HE8,&H1F,&H00,&HB4,&H12,&HCD,&H21,&H02BC
570 DATA &H3C,&HFF,&H74,&H0A,&H2E,&HFF,&H06,&H05,&H02F1
580 DATA &H01,&HE8,&H0F,&H00,&HE2,&HEE,&H2E,&HA1,&H0397
590 DATA &H05,&H01,&H8B,&H7E,&H06,&H89,&H05,&H5D,&H0200
600 DATA &HCA,&H0A,&H00,&H2E,&H8B,&H1E,&H07,&H01,&H01B3
610 DATA &H8B,&H3F,&H83,&HC3,&H03,&H2E,&H89,&H1E,&H02E8
620 DATA &H07,&H01,&H2E,&H8B,&H36,&H03,&H01,&H46,&H0141
630 DATA &HB9,&H08,&H00,&HF3,&HA4,&HC6,&H05,&H2E,&H0351
640 DATA &H47,&HB9,&H03,&H00,&HF3,&HA4,&HC3,&H00,&H035D
```

Figure 1

```
10 '      ***** THIS PROGRAM DEMONSTRATES DIRECTRY *****
20 '
30 '        **** STORE DIRECTRY SUBROUTINE IN STRING
40 '
50 KEY OFF
60 DEF SEG
70 SUBRT$ = STRING$(159,32)
80 SUBLC% = VARPTR(SUBRT$)
90 GOSUB 550
100 BLOAD "DIRECTRY",DIRECT
110 '
120 '     ** DEFINE CONSTANTS NECESSARY IN PROGRAM
130 '
140 DIM DIRLST$(111)
150 FCB$ = STRING$(33,32)
160 DTA$ = STRING$(33,32)
170 FILLER$ = STRING$(12,32)
180 COUNT% = 0
190 '
200 '     ** SET DIRLST(0) TO BLANKS
210 '
220 FOR LOOP% = 0 TO 111:DIRLST$(LOOP%) = FILLER$ : NEXT
230 '
240 CLS
250 '
260 '     ** SELECT DRIVE FOR READING DIRECTORY
270 '
280 LOCATE 1,20
290 PRINT "*** DEMONSTRATE DIRECTRY SUBROUTINE ***"
300 LOCATE 3,10,1
310 PRINT "Specify drive letter for directory (A B C D) : ";
320 GOSUB 610:DRV$ = KY$
330 '
340 '     ** FIND OUT IF DRIVE LETTER IS VALID
350 '
360 HIT% = INSTR("ABCD",DRV$)
370 IF HIT% = 0 THEN SOUND 50,7 : GOTO 300 ELSE PRINT DRV$
380 '
390 '     ** CALL TO DIRECTRY SUBROUTINE
400 '
410 LOCATE ,,0:GOSUB 550
420 CALL DIRECT (DRV$,FCB$,DTA$,DIRLST$(0),COUNT%)
430 '
440 '     ** LIST RESULTS OF DIRECTRY CALL
450 '
460 LOCATE 5,10
470 PRINT "Directory on drive " DRV$ ": contains " COUNT% " entries "
480 PRINT
490 FOR LOOP = 0 TO COUNT%-1 : PRINT DIRLST$(LOOP) SPC(6); : NEXT : PRINT
500 PRINT : PRINT "Press SPACE BAR to continue or (S) to Stop "
510 BEEP
520 GOSUB 610:CN$ = KY$
530 IF CN$ = "S" THEN END
540 GOTO 180
550 '
560 '     ** RETRIEVE LOCATION OF SUBROUTINE
570 '
580 DIRECT = PEEK(SUBLC% + 1) + PEEK(SUBLC% + 2) * 256
590 RETURN
600 '
610 '     **** KEYIN ROUTINE
620 '
630 KY$ = INKEY$ : IF KY$ = "" THEN 630
640 '
650 '     ** CHECK FOR SMALL OR CAPITAL LETTERS
660 '
670 IF KY$ < CHR$(97) OR KY$ > CHR$(122) THEN 730
680 '
690 '     ** CHANGE LOWERCASE TO UPPERCASE
700 '
710 KY$ = CHR$(ASC(KY$)-32)
720 '
730 RETURN
```

Figure 2

Directry Exposed. Figure 4 shows the commented assembly listing for the Directry subroutine, demonstrating the subroutine's three primary tasks.

First, information regarding the passed parameters must be retrieved from the stack. Next, the DOS function calls retrieve the directory entries. Finally, the subroutine returns the count of the directory entries in the Count% variable. While taking a closer look at how each of these tasks is performed, you might also refer to the DOS manual (Appendix D) to see how the hex 11, 12, and 1A function calls operate; you may also want to review the file structure of an FCB (see Appendix E).

The parameter information passed on the stack consists of the drive letter, FCB area, DTA area, Dirlst$ array address, and Count%. The first four are needed for execution of the hex 11 and 12 function calls. Specifically, the drive letter is changed to a number in line 20, and this number is placed into the FCB. Lines 26 through 28 place question marks in the name and extension area of the FCB. The DTA is set using the 1AH function call in lines 33 and 34; you'll recall that the DTA stores the directory information before it's inserted into Dirlst$. Also, the address of the string descriptor for the first element of array Dirlst$ is stored in LSTADDR. Since the string descriptors for each element of Dirlst$ are stored sequentially in memory, this address is incremented three bytes each time a directory entry is loaded into Dirlst$ (see figure 3). This updates the location where the next entry is to be stored.

Once the FCB has been set up, the program calls function 11H (line 40). Because of the global search characters in the filename, the first valid directory entry is retrieved and placed into the DTA. The routine in lines 61 through 74 is then used to move the directory entry into the

An illustration of how the DIRLST$ array string descriptors are stored in memory

Array element	1	2	3	4	5 . . .etc.
	0C03FD	0CF7FC	0CEBFC	0CDFFC	0CD3FC

each address is 3 bytes apart

Descriptor 1
0C = length
AB = low portion of address
FD = high portion of address

Figure 3

proper element of Dirlst$. Line 62 retrieves from LSTADDR the address of the string descriptor for the Dirlst$ entry that is to be used; then the entry is moved into the array, by means of a MOVSB instruction with a REP prefix, in lines 69 and 73.

All subsequent directory searches are done with the hex 12 function call (which uses information kept in a reserved area of the FCB to continue the search), and when no more entries are found in the directory, the count of directory entries retrieved is placed into Count% and control is returned to the Basic program.

Next time in the Basic/Assembly Line we'll expand on the Directry subroutine by adding a sort and producing a practical sorted directory list for any standard disk. ▲

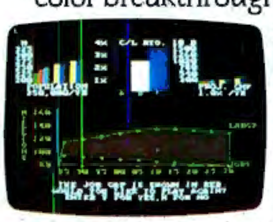
```
 1                               ; DIRECTRY
 2                               ;
 3                               ; THIS ROUTINE WILL RETRIEVE A
 4                               ;     DISKETTE'S DIRECTORY AND IS CALLED FROM BASIC
 5                               ;
 6                               ; WRITTEN BY HOWARD GLOSSER
 7                               ;
 8  0000                         CSEG        SEGMENT
 9                                           ASSUME CS:CSEG
10  0000  EB 07 90                           JMP DIRSTART
11  0003  0000                   DTADDR      DW  0                  ;ADDRESS OF DTA AREA
12  0005  0000                   DIRCNT      DW  0                  ;COUNT OF DIRECTORY ENTRIES
13  0007  0000                   LSTADDR     DW  0                  ;ADDRESS OF CURRENT DIRLST()
14  0009                         DIRECTRY    PROC    FAR
15  0009  55                     DIRSTART:   PUSH BP                ;SAVE BP FOR FAR RETURN
16  000A  8B EC                              MOV BP,SP              ;MOVE STACK POINTER TO BP
17  000C  8B 5E 0E                           MOV BX,[BP]+14         ;POINT BX AT PARM 1
18  000F  8B 77 01                           MOV SI,1[BX]           ;GET DRIVE LETTER
19  0012  8B 04                              MOV AX,[SI]            ;MOVE DRIVE LETTER TO AX
20  0014  35 0040                            XOR AX,40H             ;CONVERT LETTER TO NUMBER
21  0017  8B 5E 0C                           MOV BX,[BP]+12         ;POINT BX AT PARM 2
22  001A  8B 7F 01                           MOV DI,1[BX]           ;GET FCB ADDRESS
23  001D  88 05                              MOV BYTE PTR [DI],AL   ;PUT DRIVE NUMBER IN FCB
24  001F  57                                 PUSH DI                ;SAVE DI (FCB ADDRESS)
25  0020  47                                 INC DI                 ;BUMP DI PAST DRIVE NUMBER
26  0021  B0 3F                              MOV AL,'?'             ;PUT ? IN AL FOR STOSB
27  0023  B9 000B                            MOV CX,11              ;LENGTH OF NAME/EXT
28  0026  F3/ AA                             REP STOSB              ;FILL NAME/EXT WITH 11 '?'
29  0028  2E: C7 06 0005 R 0000              MOV CS:DIRCNT,0        ;MOVE 0 TO DIRECTRY COUNT
30  002F  8B 5E 0A                           MOV BX,[BP]+10         ;POINT BX AT PARM 3
31  0032  8B 57 01                           MOV DX,1[BX]           ;GET DTA AREA
32  0035  2E: 89 16 0003 R                   MOV CS:DTADDR,DX       ;STORE DTA ADDRESS
33  003A  B4 1A                              MOV AH,1AH             ;SET AH FOR DTA FUNCTION CALL
34  003C  CD 21                              INT 21H                ;DOS INTERRUPT
35  003E  8B 5E 08                           MOV BX,[BP]+8          ;POINT BX AT PARM 4
36  0041  2E: 89 1E 0007 R                   MOV CS:LSTADDR,BX      ;STORE DIRLST() DESCRIPTOR ADDR
37  0046  2E: FF 06 0007 R                   INC CS:LSTADDR         ;POINT TO ADDRESS, NOT LENGTH
38  004B  5A                                 POP DX                 ;PUT FCB ADDRESS IN DX
39  004C  B4 11                              MOV AH,11H             ;SET AH FOR DIRECTORY SEARCH
40  004E  CD 21                              INT 21H                ;DOS INTERRUPT
41  0050  3C FF                              CMP AL,0FFH            ;ANY MATCHES FOUND?
42  0052  74 1A                              JE DONE                ;NO - DIRECTORY IS EMPTY
43  0054  2E: FF 06 0005 R                   INC CS:DIRCNT          ;YES - COUNT FIRST ENTRY
44  0059  E8 007B R                          CALL MOVE              ;GO MOVE ENTRY TO DIRLST()
45  005C                         DIRLOOP:
46  005C  B4 12                              MOV AH,12H             ;SET AH FOR SUBSEQUENT SEARCH
47  005E  CD 21                              INT 21H                ;DOS INTERRUPT
48  0060  3C FF                              CMP AL,0FFH            ;ANY MATCHES FOUND?
49  0062  74 0A                              JE DONE                ;NO - WE ARE DONE
50  0064  2E: FF 06 0005 R                   INC CS:DIRCNT          ;YES - COUNT ENTRY
51  0069  E8 007B R                          CALL MOVE              ;GO MOVE ENTRY TO DIRLST()
52  006C  E2 EE                              LOOP DIRLOOP           ;DO AGAIN
53  006E                         DONE:
54  006E  2E: A1 0005 R                      MOV AX,CS:DIRCNT       ;MOVE ENTRIES COUNT TO AX
55  0072  8B 7E 06                           MOV DI,[BP]+6          ;GET ADDRESS OF COUNT%
56  0075  89 05                              MOV [DI],AX            ;PUT ENTRY COUNT IN COUNT%
57  0077  5D                                 POP BP                 ;RESTORE BP
58  0078  CA 000A                            RET 10                 ;FAR RETURN WITH 5 PARMS
59  007B                         DIRECTRY    ENDP
60                               ;
61  007B                         MOVE        PROC NEAR
62  007B  2E: 8B 1E 0007 R                   MOV BX,CS:LSTADDR      ;MOVE DIRLST() ADDR TO BX
63  0080  8B 3F                              MOV DI,[BX]            ;POINT DI AT DIRLST()
64  0082  83 C3 03                           ADD BX,3               ;BUMP TO NEXT DESCRIPTOR ADDRESS
65  0085  2E: 89 1E 0007 R                   MOV CS:LSTADDR,BX      ;STORE DESCRIPTOR ADDRESS
66  008A  2E: 8B 36 0003 R                   MOV SI,CS:DTADDR       ;POINT SI AT DTA AREA
67  008F  46                                 INC SI                 ;BUMP SI PAST DRIVE NUMBER
68  0090  B9 0008                            MOV CX,8               ;LENGTH OF NAME
69  0093  F3/ A4                             REP MOVSB              ;MOVE NAME INTO DIRLST()
70  0095  C6 05 2E                           MOV BYTE PTR [DI],'.'  ;PUT '.' IN FILENAME
71  0098  47                                 INC DI                 ;BUMP PAST '.'
72  0099  B9 0003                            MOV CX,3               ;LENGTH OF EXTENSION
73  009C  F3/ A4                             REP MOVSB              ;MOVE EXTENSION INTO DIRLST()
74  009E  C3                                 RET                    ;END 'MOVE' SUBROUTINE
75  009F                         MOVE        ENDP
76  009F                         CSEG        ENDS
77                                           END
```

Figure 4

COLORPLUS™ Shatters The Mold.

A vibrant, varied new world of color is now at your fingertips.

We've shattered the mold for color graphics adapters with COLORPLUS, a high resolution, full color breakthrough fully compatible with IBM Personal Computer hardware and software.



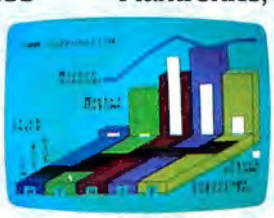
COLORPLUS produces incredibly sharp, detailed multi-color graphics of professional quality at a low PC price.* Its state of the art biplanar technology enables high resolution 4 color 80 character graphics or medium resolution 16 color 40 character graphics.

The current IBM PC capabilities of high resolution 1 color 80 character graphics and medium resolution 4 color 40 character graphics simply pale in comparison, now that the era of COLORPLUS has arrived.

Using only a single expansion slot, COLORPLUS includes an integrated parallel port, thereby saving you an expansion slot for future needs. No modification is required to existing software in the IBM modes. Enhanced software supporting COLORPLUS Graphics capabilities is currently available from major software suppliers.

XT COMPATIBLE

For over 31 years, Frederick Electronics Corporation has developed high quality electronic equipment world-wide. The key to our success is our international reputation for customer support. PC+ Products will continue in this tradition. Frederick Electronics Corporation is a subsidiary of Plantronics, Inc., a publicly owned company listed on the NYSE as PLX.



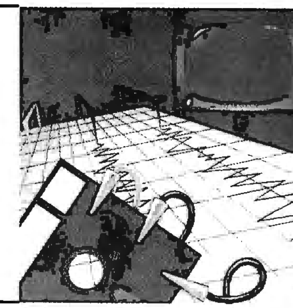
The goal of PC+ Products is to expand your options, to make your personal computer more valuable and productive. COLORPLUS is just one example of our commitment to that objective.

Contact us for the name of your nearest COLORPLUS dealer.

PC+™ PRODUCTS

PC+ Products
Plantronics Enhanced Graphic Products, Inc.
1751 McCarthy Boulevard
Milpitas, CA 95035
408 945-8711   800 638-6211
TELEX: 17-1139

marketalk reviews



Unless otherwise indicated, software listed runs in DOS 1.1 or 2.0 with either display adapter and requires 64K and at least one disk drive.

TK!Solver

TK!Solver is an equation processor. It can solve complex formulas used in such fields as science, mathematics, or engineering. But when you tune up your PC and sit down at the keyboard, *TK!Solver* is one ditty you won't play by ear: It's a complex program that has to be learned by rote.

TK!Solver uses *sheets*, or windows, to display various types of data. It also uses *VisiCalc*-style one-letter slash commands in a somewhat unfriendly fashion. The procedures, commands, and symbols rely more on mnemonics than on logic; but if you should forget a command, *TK!Solver* has a help file you can call on.

When you fire up *TK!Solver*, the PC's screen is divided into two windows, with a rule sheet for formulas at the bottom and a variables sheet at the top. As you enter the mathematical model in the rule sheet on the lower screen, the variable names are displayed in the upper screen. After you have entered the known variables, you press the exclamation point to command a solution and *TK!Solver* supplies the unknowns—in up to twelve-digit precision.

In the process of (or in addition to) solving equations, *TK!Solver* can convert units for you. If your formula uses dollars but you want results expressed in pesos, or if your variables are in square inches but you need the answer in square centimeters, *TK!Solver* will perform the conversions automatically.

You can also play the same "what if?" games that you might enjoy on a spreadsheet—testing different values to see how they affect results. Or you can enter a list of values for a variable (say, for interest rates) and have *TK!Solver* provide results for each variable on the list (show all the possible monthly payments). Such data can be displayed as tables or graphs on your screen or your printer.

TK!Solver's power lies in the equals sign, which it uses as a declaration of equality rather than as an assign statement. Unlike Basic, which must have the known values on the right side of the equals signs, *TK!Solver* can come up with answers no matter where your unknowns are placed.

TK!Solver has two methods of arriving at the answers you need. The Direct Solver is used to solve equations to which a forthright approach is possible—such as $2 * A = 6$. For the Direct Solver to work, the unknown variable can only appear once in the equation; $A * A = 4$ will boggle the Direct Solver.

When the equation is too complex for the Direct Solver, the Iterative Solver is used. The Iterative Solver starts with a guess that you provide for one of the variables and tries successive replacements until it can zero in on the answer. You can select the degree of accuracy you wish and the number of tries you want the Iterative Solver to make.

TK!Solver has built-in functions for trigonometry, logarithmic and hyperbolic functions, and for minimum, maximum, net present value, pi, and so on. And it allows you to create your own complex functions.

Although *TK!Solver* operates with its own file format, it also recognizes the DIF format pioneered by Software Arts for *VisiCalc*. *TK!Solver* can merge files and can write in print format to disk so that you can produce your hard copy later using the DOS *type* command or an editor or word processor.

Let's say that you're the industrial engineer at the Nebraska Prune Packing Company. You must design a can that will hold 36 cubic inches of Premium Pitted Prunes using the smallest quantity of tin possible.

For this task you'll need a formula for the volume of a cylinder:

Volume equals radius to the second power times pi times height

To measure the body of the can you'll need a formula for the area of a cylinder:

Area equals two times radius times height

plus the formula

two times radius to the second power times pi

to include the top and bottom.

Here's how to set that up. In the rule sheet, enter the formulas

$$V = r^2 * PI * h$$

and

$$A = 2 * r * h + 2 * (r^2 * PI)$$

As you enter these formulas (in the lower sheet), the variables are duplicated automatically in the upper sheet. When you type a semicolon, the cursor jumps from the bottom sheet to the top sheet, where you enter those variables that you already know—such as 36, the volume of the can, and the value 3.1416 for pi. You put this data in the input column. You'll note that the top sheet includes an *S* column, and that this column has an asterisk in it. This is a status column; the asterisk indicates that the problem isn't yet solved. You'll also find a remarks column, in which you may add comments so at some future date you won't confuse your Prune formula with your Pickle formula.

Now you're ready to play "what if?" by entering a value for one of the other variables and typing *!* to summon a solution; the solution will appear in the output column. You'll find that a radius of 1 results in a can almost a foot tall. This can would use up over 78 square inches of tin—enough to choke a goat. Play it again with a radius of 2 and your tin requirement drops to a scant 61 square inches.

But there's an easier way to test the various possible radii. You could leap to the list sheet, place a *1* in the top row, and scroll down to the eleventh row to insert a *3*. *TK!Solver* will fill in the blanks with values at increments of .2. Back in the variables sheet, typing an *L* in the status column will tell *TK!Solver* to solve the entire list of values for you.

You'll note that our formulas used radius to the second power rather than radius times radius; this approach lets us use the Direct Solver.

Now suppose you want to see if you can make a can with 60.5 square inches of material.

You have a value to input for the can's area, but since there will be two unknowns, you can't use the Direct Solver. Entering *G* in the status column for the radius tells *TK!Solver* that you want guesses; this is its cue to call on the Iterative Solver. The Iterative Solver will substitute a value for your radius and increase or decrease it as necessary until it zeros in on your answer. In this example, you can make a can using 60.5 square inches of tin with a radius of 1.8783835393 and a height of 3.2477533476 inches.

There's just one chore left. Some of your Premium Prunes are destined for the export market, and for the Albanian trade you have to express the volume in *zods* instead of cubic inches. Enter the conversion formula in the unit sheet, and the measure you specify will be used for computation and measurement. You can go on to the unit subsheet and

designate a display unit that is different from the calculation unit, thus telling *TK!Solver* to compute in cubic inches and display in zods.

Now you are ready to print a copy of your results or send the file to disk for later use. The job is done.

Designing the Premium Prune can help show that *TK!Solver* is a complex program; its intricate design isn't suitable for simple problems. However, as the complexity of your formulas increase, the value of this program grows immensely.

Software Arts gives you everything you need to go with *TK!Solver*—except a bumper sticker. The documentation fills a chubby three-ring binder with a brief introductory guide, an instruction manual, and a thumb-tabbed reference manual complete with glossary and index. There is a quick-reference card and a wall chart to help you with your first steps. The disk is copy-protected, but one backup is provided.

The folks at Software Arts are eager to give *TK!Solver* all the support it needs. They have just provided a free upgrade to registered owners; the new version (1.2) supports DOS 2.0 and hard disk. The company also provides one free issue of *Software Arts Technical Notes (TK!SATN)*, a bimonthly journal that offers tips for *TK!Solver* owners.

Also available from the same vendor are *TK!SolverPacks*, which provide canned applications for specific fields. So far, there are packages for mechanical engineering, financial management, and introductory science. Each contains about a dozen mathematical models.

In sum, *TK!Solver* is a powerful but difficult program. Part of its complexity lies in the number of separate sheets that must be used. There's one for rules or formulas and another for variables. There's one to contain the units of measure and the conversions to other units. There are global sheets and list sheets, table sheets and plot sheets. There's even a sheet for user-defined functions. What's more, some of these sheets have their own subsheets for you to contend with.

The graphs as displayed or printed depict the data clearly enough, but they are rather crude (they're done with text characters and hence do not require a color/graphics adapter). But *TK!Solver*'s purpose is to provide answers rather than pretty pictures, and it solves complex financial and engineering formulas that would choke any spreadsheet or Basic program.

If you're a professional who has to deal with the same complex formulas repeatedly, you may find *TK!Solver* a great improvement over working with a calculator. But if you're looking for something to help your kid with math homework, steer clear of this complex program.

The *TK* in *TK!Solver* stands for tool kit, and the people at Software Arts seem to think it's as essential in your computer as those Ginsu knives in your kitchen drawer. If you're a professional number juggler, they're right. DR

TK!Solver, Software Arts (27 Mica Lane, Wellesley, MA 02181; 617-237-4000). Requires 96K. $299.

The Witness

Everything about *The Witness* reflects the style of the pulp detective novel of the late thirties and early forties—history, rather than science fiction, from our perspective. And all the era-relevant details of the game and the package have been carefully researched for accuracy.

The package itself is fun. Cliche after cliche from those popular pulps jumps off the outside folder in thirties-traditional multiple typefaces. Raymond Chandler and Dashiell Hammett, move over; here comes Infocom. The dossier contains a bunch of clues, including a pack of matches from the Brass Lantern restaurant; a thirties-style telegram actually set up and authenticated by Western Union; a page from the *Santa Ana Register* of February 1, 1938, doctored only to include the story relevant to the game; and a copy of *Nat'l Detective Gazette*, complete with authentic ads from the *National Sheriff* of 1938 and filled with stories that supply the documentation for the game.

The Witness is fun to play, but the mystery adventure form is still in its infancy. Although the interaction is terrific, it's kind of like terrific caviar; it makes you hunger for the main course. You'll think of a hundred questions you know you can't ask, and it's not totally consoling knowing that if you can't ask them you don't need them. Still, if this first step weren't so good, it wouldn't make you think of how much more you'd like.

The mystery, when all is said and done, is fairly simple; *Deadline* was more complex. But the atmosphere and sense of reality make up for it. *The Witness* is another plum for mystery buffs. And, like the pulps it emulates, the entire package is apt to become a collector's item some day. MCT

The Witness, by Steve Galley, Infocom (55 Wheeler Street, Cambridge, MA 02138; 617-492-1031). $49.95.

Ultralight Command

Ultralight Command is a high-speed, horizontal scrolling shoot-'em-up type game with a real arcade feel. You command an ultralight aircraft that patrols a bright blue expanse of water, protecting a small rowboat and its inhabitants from three types of bomb-laden aircraft coming at you from both sides of the screen.

One of the most thrilling aspects of playing a scrolling game is the ability to maneuver off screen, to the left and the right, guarding a much larger space than appears at any one time on the screen. You patrol a terrain that seems to go on forever. *U.C.* takes this thrill one step farther by giving you the ability to follow enemy ships and encounter them off screen. In other home arcade scrolling games, off screen is off limits. Of course, in *U.C.*, you don't actually go off screen, but it's a powerful illusion just the same.

The sound for this game is equally compelling. As the excitement mounts and you finish a successful round, a crisp voice calls out "attack wave completed." At the same time, beeps, whoops, and whimsical

OPEN UP AN ACCOUNTING DEPARTMENT FOR $395.00

GENERAL LEDGER integrated postings from A/R, A/P and Payroll. **Prints 13 detailed reports** • Company or departmental Income Statements • **Comparative financial statements** with current, YTD, budget, and last year (month and YTD) • Presents everything you, your bookkeeper, and your accountant need to know • G/L reconciles all accounts and maintains extensive, detailed audit trails • Trial Balance includes all transactions • **Flexible Chart of Accounts** • True double-entry bookkeeping • Master File capacity: 400 accounts • Monthly Transaction capacity: 1,000 with 200K diskette; 3,500 with 500K diskette; 7,000 per Megabyte with a hard disk.

ACCOUNTS RECEIVABLE provides **instant, on-line customer account information (both current and aged)**, complete invoicing (open-item or balance forward) and statement capabilities on optional preprinted forms give your company a professional image • Quickly identify overdue accounts, speed collections, help control cash flow • Detailed and summary customer activity and aging reports • Produces 8 reports • Automatic periodic customer/client billing option • Itemized monthly transactions • Master File capacity: 400 Customers • Monthly Transactions capacity: 800 with 200K diskette; 3,500 with 500K diskette; 7,000 per Megabyte with a hard disk.

ACCOUNTS PAYABLE maintains complete vendor/voucher history and includes **check-writing capabilities** • **Current and aged payable reports** • Cash flow/cash requirements report • Prints checks with comprehensive check stubs • **Produces 11 reports and documents** • Automatic pay selection program allows payment by due date or by discount date • Manual and automatic checkwriting • Check register • Master File capacity: 400 Vendors • Monthly Transactions capacity: 800 with 200K diskette; 3,500 with 500K diskette; 7,000 per Megabyte with a hard disk.

PAYROLL—Be the office hero each week when the checks come out on time! • **Calculates payroll for every type of employee** (hourly, salaried, and commissioned) and **prints payroll checks** (with popular, comprehensive check stubs) with an absolute minimum of input • Maintains monthly, quarterly, and yearly totals for reporting in **multiple states** • **User-maintainable Federal, State, and local tax tables** • W-2 printing • 941 Reporting • Produces 10 reports • Master File capacity: 400 employees.

Why staff up? With the Desktop Accountant™, all the accounting help your office needs can be at your fingertips!

Open up a wide new range of possibilities for your microcomputer! No matter what type of business you're in, Desktop Accountant will let you manage the financial end of it more professionally than ever before.

A Complete System with Support. Desktop Accountant includes General Ledger, Accounts Receivable, Accounts Payable and Payroll programs, along with comprehensive user manuals and training aids. We've even prepared an audio cassette tape to make learning the system fast and fun. And our telephone "hotline" means personalized support whenever you need it.

Desktop Accountant's fully integrated accounting system is a complete package of software, training aids, manuals and user's newsletter.

Produces 42 Reports. "Keeping the books" has never been so easy! Desktop Accountant prepares every bookkeeping and accounting report your growing business requires: from invoicing to statements to aged A/R listings; from cash distribution to A/P checks to vendor activity reports; from **complete** payroll checks and stubs to W-2 forms; from the chart of accounts to balance sheet and income statement, as well as many others so vital to efficient management.

Desktop Accountant is available for nearly every portable, personal and desktop computer. The system requires either CP/M® or MS-DOS™ (PC-DOS), Microsoft BASIC™, 64K RAM, two disk drives or hard disk, and a 132-column printer (or an 8½" x 11" printer with compressed print mode).

You won't find better quality software at such a low price—a price we can offer now because development costs were recovered years ago. Just **$395.00 for most CP/M® formats ($495.00 for IBM® and some CP/M formats) complete.** *Call for available formats.*

Order Desktop Accountant today! You'll soon see why we call it **INTELLIGENT SOFTWARE FOR INTELLIGENT PEOPLE.**

ROCKY MOUNTAIN SOFTWARE SYSTEMS
Business Microcomputers and Software

• California residents add 6½% Sales Tax • Payment by VISA/MasterCard/COD/MO/Cashier's Check • All Brand Names are manufacturers' registered trademarks • No sales to Dealers • Foreign orders please call or write before ordering • ©1983 Rocky Mountain Software Systems.

MasterCard VISA

To order Desktop Accountant and for more comprehensive literature, call toll-free:

☎1-800-832-2244
(In California call 1-800-732-2311)

or send orders to:
DESKTOP ACCOUNTANT
1280-C Newell Avenue, Suite 1219
Walnut Creek, CA 94596

music collide with eye-popping visuals. As the game action speeds up, you're transported into a bright arcade of the mind.

The only complaint you might have about *Ultralight Command* is that it is too hard on the eyes, with a red border on the screen framing a deep blue sea. Also, the fact that you get an extra ship every 2,000 points doesn't seem much of a challenge.

But fun is fun, and that's what *U.C.* is.

Although it's not a children's game, kids will love *Ultralight Command.* The fun is addicting, and the first level of difficulty is not hard to master. Fast and furious when played at full throttle, levels 2 and 3 will give experienced players more than a run for their money. MF

Ultralight Command, by Gerald Seagraves. Fast N Fun Video (1074 East Sandpiper Drive, Tempe, AZ 85253; 602-831-9363). Requires color/graphics adapter. $39.95.

Checkbase

This "personal financial control" program maintains a check register, keeps an up-to-date balance, and prints checks. Checks can be listed by date, reconciliation status, payee, and a number of other choices and combinations. Lists can be printed with the user's choice of title.

Checkbase is designed to handle multiple checkbooks, and it prompts the user for the name of the checkbook being updated before data entry begins. An automatic date function lets you use the current system date as the check date. Each time a check is created, the program asks if it should be printed. If the answer is yes, the user is directed to align the printer and printing is done immediately. If the answer is no, the user is prompted for the next check number. This simplifies conversion from a standard manual system, because checks that have already been issued can be added to the database without being duplicated. There is, however, no way to go back and print a check if printing was skipped when the check was created.

The print format is flexible, allowing the user to set tab positions for the date, payee, numeric amount, and written amount. A test mode included in the program prints an alignment pattern according to the specified tab positions. The year can be either four or two digits, and the tab position of the two-digit field is user-controlled.

The print program is intended for a parallel printer but will work on serial units if the output is redirected via the *mode* command.

An embedded trap checks for duplicate check numbers, but there is no way to run an exception report showing missing checks. A check can be voided, which leaves the check number in the register with an amount of zero; checks also can be deleted.

The running balance displayed at the bottom of the screen is debited automatically each time a check is created. The user is asked if the check has been reconciled with a bank statement; if it has, the information is added to the record. Reconciliations can be done later as well.

No provision exists for adding a comment or memo, which would be useful for tax purposes, but there is a "tax category" field that allows the user to enter a single character. You can use this field as a way of categorizing tax-deductible items; your medical expenses, for example, could be coded with an *M* and listed separately from your charitable contributions, interest expense, and so on. Any keyboard character except the comma can be used as a category code, and the system distinguishes between upper and lowercase characters.

Although *Checkbase* appears to be intended for the individual user, some of its features—check printing, for example—seem more applicable to a small business. Inserting one check at a time in the printer and attempting to line it up seems like more trouble than most individual users will want to take. *Checkbase* has no batch mode, so there's no way to print more than one check at a time; batch capability would make this package more attractive to small businesses.

The user manual is easy to understand and contains a tutorial that


25 Mb the hard way.
25 Mb the Rana way.
© 1983 Rana Systems

explains exactly how to do credit and debit entries and reconciliations. There is no index, but the table of contents is arranged by function, and information can be found easily.

In short, *Checkbase* is a good solid register and reporting system. Its list functions can provide excellent tracking of budget expenses, and those tax category codes take a big step toward easing the April 15 blues. It's easy to use, requires little effort to learn, and will be a valuable tool for anyone who writes many checks. DC

Checkbase, International Microcomputer Software (633 Fifth Avenue, San Rafael, CA 94901; 415-454-7101). $95.

Rogue

Milo was sweating heavily in his armor. A trickle of blood from the snake bite on his left ankle was gradually filling his boot. He squinted at the darkness, nervously shifting his mace from hand to hand.

Milo hated the dark. He hated the humidity, the slimy walls, the silence. He hated everything about this creepy labyrinth. Damn the guildmasters! Let them find their own cursed amulet. What right had they to send him here to die like the others?

They hadn't told him about the others. But now, when it was too late, he kept finding them. Or what was left of them. Mostly just a scattering of bones on the cavern floor. Sometimes he could make out the remains of a suit of armor, like the one the guildmasters had given him, or a mace like the one in his right hand. One of the corpses, more prosperous than the rest, had owned a silver wand. It was now in Milo's pack.

Wait! What was that? Something came shuffling out of the darkness. An enormous troll! He couldn't fight that!

As the troll swung at him, Milo reached for the silver wand. He flourished it with a gesture he had once seen a wizard use.

The troll was gone! There was just a big plant sitting there, viney-looking, with a big pulpy flower. Milo smiled with satisfaction. With this great wand, he was invincible.

Then a thought troubled him. If the original owner had had this great wand, how come he was dead?

If you have played *Dungeons & Dragons* or the like, Milo's story will seem familiar. If not, you may not understand the attractions of finding gold and slaying monsters in the dank confines of a labyrinthine dungeon. Suffice to say, this kind of game has been popular for about ten years. It can be addictive, especially for younger players.

Rogue was originally written under Unix and has now been adapted for PC-DOS.

You start on level 1 of the dungeon with twelve hit points and sixteen strength points. You are wearing a suit of +1 ring mail. You also have a magical mace, a +1 bow, and some depressingly normal arrows. Your most important piece of equipment, however, is the sandwich in your pocket. You can fight with your bare hands if necessary, but there's no substitute for food, and starvation is a major threat.

From this standing start, you can proceed to scrounge up the really interesting stuff—wands, potions, magic scrolls, rings of protection, improved weaponry—all scattered around rather liberally (this is what gamers sometimes call a "gilded hole").

Offsetting this bounty is an alphabet of monsters, from aquator to zombie. Each has a special modus operandi. An ice monster will try to numb you, an aquator will rust away your armor, a wraith will steal your hit points. A nymph will smile seductively and try to pick your pocket. As you kill the monsters, you progress in your trade as fighter, increasing the number of hits you can absorb.

You also accumulate valuable knowledge. The various treasures in the dungeon are hard to identify. A staff of light, for example, may be described simply as "a banyan staff." This changes from game to game, so "a red potion" may heal you in one game and poison you in the next. As each game progresses, you accumulate a large number of items, some tried and true, others unknown and possibly cursed. Hitting the D key at any point displays the knowledge you have accumulated, so you can make clever little decisions like, "I know this scroll labeled 'orgevsakspo' isn't the one that fries brains, so it's probably safe to read it."

Everything appears on the screen as characters from the standard set. Your hero is a "happy face," the monsters are represented by letters, the dungeon walls are simple lines, and so on. Although the graphics aren't very exciting (needless to say), this feature means the game looks equally good with either display adapter.

The only things that show up on the display are the ones your character knows about. Some rooms are brightly lit, but as you descend, more and more are dark. This creates an atmosphere of mystery as your character creeps around with one hand to the wall, not knowing if a monster is waiting to pounce on him. To heighten this effect, there are traps and secret doors that are invisible unless you stop and search for them. Some of the monsters are invisible, too. The cumulative effect is quite spooky, like being in a strange house at night or driving in a thick fog.

Unfortunately, the game has some less admirable complications. Your possessions, for example, are identified by letter, and the letters keep changing. As a result, throwing item F might mean shooting an arrow one minute and discarding a valuable healing potion the next. Also, it is annoying that the monsters don't seem to be affected by the traps. However, the biggest problem with this game is that it's so hard to stop playing. The maze is enormous, and even when you're killed you're inclined to try again.

Milo gasped in horror as the plant's tendril wrapped itself hungrily around his leg . . . . FJ

Rogue, by Michael C. Toy, Kenneth C.R.C. Arnold, and Jon Lane. AIDesign Systems (1989 Santa Cruz Avenue, Santa Clara, CA 95051; 408-296-1634). Requires 128K. $44.95.

Enchanter

Where is Zork? Alive and well in magic school. The result? *Enchanter*, the beginning of a trilogy sequel to the *Zorks*. No longer is Zork condemned to subterranea. *Enchanter* takes place in a natural—if fanciful—world, a sunlit world. Well, at first it's a sunlit world. But the powers of darkness would have it otherwise. As the apprentice enchanter pursues his quest, the days grow alarmingly shorter. Soon the sun is rising at noon and setting at 2:00 p.m. Only the young apprentice's quick study and apt use of powerful magic can save the world from perpetual night.

Can magic be logical? Certainly. Determining when and where and how to use which spells provides puzzles as clever and logical as those in *Zork*. Without clear, innovative thinking, you will find the enchanter's magic is useless. Or sometimes when it's useful it's misguided. Several spells can be cast only once; exactly the same number of puzzles can be solved only one way—with those spells. But the spells work in plenty of other places and even solve other puzzles that seem to be insoluble otherwise.

In line with Infocom's new mysteries and recent science-fiction adventures, *Enchanter* expands interaction with other characters beyond that of earlier Infocom games and of most other games. Even animals become confidants—or informers—when you have the right magic.

Enchanter is not chock-full of breakthroughs. It is simply a delightful adventure that takes advantage of the niceties and expertise Infocom has been nurturing at such a fast clip. Zork's world isn't forgotten. The period is the same, or very close to it, and, although these people live above ground, they remember Zork even if you don't. Check the portraits in the castle gallery; see yourself azorking in the hall of mirrors.

And don't make friends with dragons. MCT

Enchanter, by Marc Blanc and Dave Lebling, Infocom (55 Wheeler Street, Cambridge, MA 02138; 617-492-1031). $49.95. ▲

THE PROFIT PLOT

by Jack Grushcow

A Financial Statement Analysis Template

For the past two months, we've been discussing financial ratios in the context of a hypothetical widget manufacturing company. This month we jump boldly into the real world. We'll apply these same ratios to none other than Big Blue Itself, and we'll build a financial statement analysis spreadsheet around these ratios.

Our spreadsheet will paint a detailed financial portrait of IBM; of course, you can also use the same spreadsheet design to examine any other company. We'll calculate thirteen financial ratios, describing Big Blue's health in the areas of liquidity, long-term solvency, efficiency, and profitability. In case you've just joined us, you can find a complete description of these business ratios and their interpretation in the November and December installments of this column.

As you'll see by looking at figure 1, our spreadsheet consists of four main sections. The first section, at the top of the model, holds the income statement information. The second, beginning at row 41, holds the statement of retained earnings. The third section is used to record the balance sheet data; it begins at row 54. And the final section, beginning at row 135, is for analysis. All the ratios appear in this section, ordered by ratio type.

Familiarize yourself with the spreadsheet layout. If you know where to look for the various types of financial data, you'll find it much easier to insert the information from a corporate financial statement into your own spreadsheet.

Row 1	INCOME STATEMENT
Row 41	STATEMENT OF RETAINED EARNINGS
Row 54	BALANCE SHEET
Row 135	ANALYSIS SECTION

SPREADSHEET LAYOUT DIAGRAM

This spreadsheet design can accommodate up to ten financial statements—one per column. That means you can choose either to analyze a single company's performance over time or to perform a comparative analysis of several different firms in the same industry.

If you're an investor, you may want to keep a separate model for each company you've invested in, updating the sheet annually or quarterly as the financial information becomes available. This would allow you to monitor important trends in a company's performance.

Suppose the financial press is projecting a 15 percent growth in a company's earnings. You can enter this information and see what impact this rate of growth will have on other areas of the company. By putting several competitive companies on one sheet, you can monitor industry-wide patterns.

If you work in a credit department, you might want to keep a sheet for each key customer. As new financial information becomes available, you could update the spreadsheet and see if there are negative changes in any of the key ratios.

If you're a manager, you might want to use this model to track annual or quarterly results for your own company. This will give you an opportunity to foresee problems and suggest solutions.

Figure 2 shows the formula structure for our model (the first few columns only), as created in *VisiCalc*. You can expand the model as much as you like by using the copy or replicate commands for your spreadsheet program.

As you look down the rows of this model, you'll notice two features. Border rows composed of dashes appear from time to time, and some of the labels in column A begin with an asterisk. The border lines are used to enclose areas where lines can be either added or deleted. For example, the lines in rows 7 and 10 enclose income information. You can put as many categories of income as you wish between these two dashed borders.

Do not delete the total line that appears in row 11 of figure 2, nor any of the other total lines. Each of these total lines contains a formula that sums the numbers contained between these lines.

The lines that begin with asterisks are special lines containing formulas. Don't enter any data into these rows, or you'll lose your formulas. These rows will automatically display the correct amounts as you enter a company's financial information. If your spreadsheet program has cell-protection capability, use it for these rows of formulas.

To use this spreadsheet, you have to enter the financial data of the company or companies you wish to examine. Unfortunately, there's no standard format that all companies follow in the presentation of their financial reports. Companies even use different names to describe the same types of statements. For example, "statement of profit and loss," "earnings statement," and "income statement" are all different ways of

naming the same report. There's no easy way around this financial tower of Babel.

To get some practice using this spreadsheet, we'll now enter the financial information for IBM for the years 1978 through 1982. This example is fairly complex and is designed to give the model a good workout. If you're using it to examine a small private company, you won't find the model as difficult to use as it may appear in this example.

The financial statements for IBM are shown in figure 3, and our completed model appears in figure 4.

You'll notice that IBM's financial statements appear considerably more complex than our widget example of the last two months. The accounting requirements of a multibillion-dollar, multinational company obviously differ from those of a one-product manufacturer.

Rather than being intimidated by the apparent complexity of these statements, let's just dive in and start entering data. We'll make any necessary modifications as we go along. To speed up the data-entry process, turn off your spreadsheet program's automatic recalculation.

We'll begin with the income statement. Enter the company's name by moving the cursor to A1 and typing IBM. Next move the cursor to C6 and enter the years; put 1982 in column C and move backward from there so that 1978 appears in column G.

The first row of data to enter is NET SALES in row 8. You'll notice in figure 3 that IBM's revenue is divided into two categories—"Sales" and "Rentals & Services." We won't bother breaking it down to this level of detail.

Move your cursor to C8 and enter the total sales data from figure 3. Enter 34364000, 29070000, 26213000, 22862776, and 21076089 in cells C8 through C12. Note that all the figures are expressed in thousands of dollars.

The next item in the model is OTHER INCOME. In IBM's income statement, this information appears after all the expense data and above "Inc. bef. inc. taxes." Enter 328000, 368000, 430000, 449295, and 411808 in cells C9 through C12.

Now on to COST OF GOODS SOLD. IBM's income statement divides this figure into "Cost of sales" and "Cost of rentals & services." We'll put the sum of these two figures in our spreadsheet.

At C17 enter 6682000+7006000. Your spreadsheet program should display 13688000 as the total cost of goods sold for 1982. Continue in the same fashion for the other four years. When you've finished entering the data for row 17, your model should display the following totals: 13688000 for 1982, 12016000 for 1981, 10149000 for 1980, 8412958 for 1979, and 7484025 for 1978.

If you now force a manual recalculation, you'll notice that the GROSS PROFIT figure has been calculated for you in row 18. We'll use

	-A-	: -B-	: -C-	: -D-	: -E-
2			"FINANCIAL	" STATEMEN	"T ANALYSIS
3	"INCOME ST	"ATEMENT			
4					
5	"I N C O M	"E			
6	-*	-*			
7	-	-	-	/-	-
8	"NET SALES				
9	"OTHER INC	"OME			
10	/-	-	/-	-	-
11	"TOTAL INC	"OME	@SUM(C7...C10)	@SUM(D7...D10)	@SUM(E7...E10)
12	/-	/-	/-	/-	-
13					
14	"E X P E N	"S E S			
15	/-*	/-*			
16					
17	"COST OF G	"OODS SOLD			
18	"* GROSS P	"ROFIT	+C11-C17	+D11-D17	+E11-E17
19	-	/-	-	-	-
20	"GEN. SALE	"S & ADMIN			
21	"DEPRECIAT	"ION			
22	"RESEARCH	"& DEVEL.			
23	-	-	-	-	-
24	"TOTAL OPE	"RATING EX	@SUM(C19...C23)	@SUM(D19...D23)	@SUM(E19...E23)
25	/-	/-	/-	/-	/-
26	"* NET OPE	"R. INCOME	+C18-C24	+D18-D24	+E18-E24
27	"INTEREST	"EXPENSE			
28	"* INCOME	"BEF. TAX	+C26-C27	+D26-D27	+E26-E27
29	"TAXES				
30	"* NET BEF	" MIN & EX	+C28-C29	+D28-D29	+E28-E29
31	"MINORITY	"INTEREST			
32	"EXTRAORDI	"NARY ITEM			
33	"DISCONTIN	"UED ITEM			
34					
35	"* NET INC	"OME	+C30+C31+C32+C33	+D30+D31+D32+D33	+E30+E31+E32+E33
36					
37	"* PREF SH	" (000S)			
38	"* COM. SH	" (000S)			
39	"COM. MARK	"ET PRICE			
40	/-*	/-*	/-*	-*	/-*
41	"STATEMENT	" OF RETAI	"NED EARNI	"NGS	
42					
43	"RETAINED	"INCOME			
44	"*ANNUAL N	"ET INCOME	+C35	+D35	+E35
45	"*TOT RET.	" INCOME	+C43+C44	+D43+D44	+E43+E44
46					
47	"DIVIDENDS	" PAID:			
48	"PREFERRED				
49	"COMMON				
50	"* TOTAL D	"IVIDENDS	+C48+C49	+D48+D49	+E48+E49
51					
52	"* BALANCE	" YEAR END	+C45-C50	+D45-D50	+E45-E50
53	/-*	/-*	/-*	/-*	/-*
54	"BALANCE S	"HEET			
55					
56	"CURRENT A	"SSETS:			
57	/-	/-	/-	/-	/-
58	"CASH				
59	"SECURITIE	"S			
60	"RECEIVABL	"ES			
61	"INVENTORY				
62	"DEFERRED	"TAXES			
63	"OTHER CUR	". ASSETS			
64	-	/-	/-	-	/-
65	"TOTAL CUR	"RENT	@SUM(C57...C64)	@SUM(D57...D64)	@SUM(E57...E64)
66	/-	/-	/-	/-	/-
67					
68	"FIXED ASS	"ETS:			
69	-	-	/-	/-	/-
70	"PROPERTY	"PLANT,EQU			
71	"LESS ACC.	" DEPRECIAT			
72	"OTHER FIX	"ED ASSETS			
73	/-	/-	/-	/-	/-
74	"TOTAL FIX	"ED ASSETS	@SUM(C69...C73)	@SUM(D69...D73)	@SUM(E69...E73)
75	/-	-	/-	/-	-
76					
77	"INTANGIBL	"E ASSETS:			
78	/-	/-	-	-	/-
79	"GOODWILL				
80	"OTHER				
81	/-	/-	/-	/-	/-
82	"TOTAL INT	". ASSETS	@SUM(C78...C81)	@SUM(D78...D81)	@SUM(E78...E81)
83	/-	/-	/-	/-	/-
84	"* TOTAL A	"SSETS	+C65+C74+C82	+D65+D74+D82	+E65+E74+E82
85					
86					
87	"CURR. LIA	"BILITIES:			
88	/-	/-	/-	/-	/-
89	"ACCOUNTS	"PAYABLE			
90	"NOTES PAY	"ABLE			
91	"LOANS PAY	"ABLE			
92	"INCOME TA	"X PAYABLE			
93	/-	/-	/-	/-	/-
94	"TOTAL CUR	"RENT L.	@SUM(C88.C93)	@SUM(D88...D93)	@SUM(E88...E93)
95	/-	/-	/-	/-	/-
96					
97	"DEBT:				
98	/-	/-	/-	/-	/-
99	"LONG-TERM	" DEBT			
100	"OTHER L.T	". DEBT			
101	"OTHER L.T	". DEBT			
102	/-	/-	/-	/-	/-
103	"TOTAL L.T	". DEBT	@SUM(C98...C102)	@SUM(D98...D102)	@SUM(E98...E102)
104	/-	/-	/-	/-	/-
105					
106	"OTHER LIA	"BILITIES			
107	/-	/-	/-	/-	/-
108	"DEFERRED	"TAXES			
109	"OTHER LIA	"BILITIES			
110	"OTHER LIA	"BILITIES			
111	/-	/-	/-	/-	/-
112	"TOTAL OTH	"ER	@SUM(C107...C111)	@SUM(D107...D111)	@SUM(E107...E111)
113	/-	/-	/-	-	/-
114	"*TOTAL L.	"T. LIABIL	+C103+C112	+D103+D112	+E103+E112
115					
116	"*TOTAL LI	"ABILITIES	+C94+C114	+D94+D114	+E94+E114
117					
118					
119	"OWNERS EQ	"UITY:			
120	/-	/-	/-	/-	/-
121	"PREFERRED				
122	"PREFERRED				
123	"COMMON				
124	"COMMON				
125	"CAPITAL S	"URPLUS			
126	"RETAINED	"EARNINGS			
127	"LESS TREA	"S. STOCK			
128	/-	/-	/-	-	/-
129	"TOT. OWNE	"RS EQUITY	@SUM(C120...C128)	@SUM(D120...D128)	@SUM(E120...E128)
130	/-	/-	/-	/-	/-
131					
132	"*TOT LIA.	"& OWN EQU	+C116+C129	+D116+D129	+E116+E129
133					
134	/-*	/-*	/-*	/-*	/-*
135	"ANALYSIS	"SECTION			
136					
137	"LIQUIDITY	":			
138	"WORKING C	"APITAL	(C65/C94)	(D65/D94)	(E65/E94)
139	"QUICK ASS	"ET	(C65-C61)/C94	(D65-D61)/D94	(E65-E61)/E94
140					
141	"L.T. SOLV	"ENCY:			
142	"DEBT-EQUI	"TY	+C103/C129	+D103/D129	+E103/E129
143	"TIMES INT	". EARNED	+C28/C27	+D28/D27	+E28/E27
144	"ASSET COV	"ERAGE	(C84-C82-C94-C108)/C103	(D84-D82-D94-D108)/D103	(E84-E82-E94-E108) E103
145					
146	"EFFICIENC	"Y			
147	"INVENTORY	" TURNOVER	+C17/C61	+D17/D61	+E17/E61
148	"ACCTS. RE	"C. TURNOV	(C60*365)/C8	(D60*365)/D8	(E60*365)/E8
149	"SALES/WOR	"KING CAP.	+C8/(C65-C94)	+D8/(D65-D94)	+E8/(E65-E94)
150	"SPECIFIC				
151					
152	"PROFITABI	"LITY			
153	"GROSS PRO	"F. MARGIN	((C8-(C17+C21))/C8)*100	((D8-(D17+D21))/D8)*100	((E8-(E17+E21))/E8)*100
154	"NET PROF.	" MARGIN	(C30/C8)*100	(D30/D8)*100	(E30/E8)*100
155	"RETURN ON	" COMMON	((C30+C31-C48)/ (C129-C121-C122))*100	((D30+D31-D48)/ (D129-D121-D122))*100	((E30+E31-E48)/ (E129-E121-E122))*100
156	"EARNINGS	"PER SHARE	(C30+C31-C48)/C38	(D30+D31-D48)/D38	(E30+E31-E48)/E38
157	"PRICE/EAR	"NINGS	+C39/C156	+D39/D156	+E39/E156
158	/-*	/-*	/-*	/-*	/-*

Figure 2. File ratios.

this figure later in the analysis section to calculate the gross profit margin. Remember that the asterisk in the label serves as a warning; don't enter data in or delete this line, or you'll wipe out a valuable formula.

A good way to verify the data you enter is to force a recalculation and check that the totals calculated on your spreadsheet match the ones on the financial statement. Verifying that the gross profit figures match is a good example of this procedure.

We can now enter the GEN. SALES & ADMIN expenses. Looking at figure 3, we see that IBM has lumped its development and engineering expenses together with these other expenses. Our task is easier, therefore, because we won't have to enter R&D expenses separately. In row 20, columns C through G, enter 12620000, 11027000, 10324000, 9205367, and 8151129.

We don't need to put anything in the next two rows. IBM's depreciation expenses have already been included in the various cost categories, and we have already included R&D with the other general expenses. If you wanted to clean up the appearance of your income statement, you could delete rows 21 and 22. We won't do this here because we want to be able to refer to the line numbers in figure 1.

Next, enter the values shown across from "Interest charges" into the appropriate positions in the row labeled INTEREST EXPENSE. The net income before taxes is shown in line 28. Recalculate and check your figures against the totals shown in IBM's statement.

Now enter the values shown next to "U.S. Fed. & non-U.S. income taxes" in row 29, TAXES. In the next row, NET BEF MIN & EX, these taxes are deducted from net income before tax.

Since IBM's statement shows no amount for minority interest, extraordinary items, or discontinued operations, we can now move the cursor to C38. This is where we'll enter the number of common shares outstanding. In IBM's original financial statements, two figures were given: the number of common shares outstanding at year's end and the average number of common shares outstanding throughout the year. We will use the year-end figures in our financial statement analysis.

In cells C38 through G38, enter the number of common shares for each year. Since all of our money figures are in thousands, we must express the number of shares in thousands as well. So note the decimal point. Enter 602406.128, 592293.624, 583806.832, 583594.543, 583241.454 in these five cells. To keep column width under control (remember, this is *VisiCalc* and we don't have variable column widths), figure 4 shows these share values expressed to only one decimal place; for ratio calculations, however, the full three-place precision will be used.

In the next line we need to enter the market price of IBM's common stock. Here again we have a decision to make. Stock prices move daily, fluctuating considerably over the course of a year. It would be unreasonable to use the year-end stock price, because seasonal factors may have produced a biased value for the stock at that time of year. For instance, IBM's stock prices are frequently depressed at the end of the company's fiscal year (December 31), as a result of year-end tax-loss selling.

What share price to use depends on your objectives. If you're an investor deciding whether to add a stock to your portfolio, you will want to use the current price and compare it with the most recent financial information available. In the model we're building now, we're doing a five-year historical analysis, so we want to use an average share price for each year.

For demonstration purposes, we'll use a simple average of the yearly low and high stock prices. If you were doing a serious analysis, you'd probably feel more comfortable using a weighted average price. We'll discuss weighted and other special averages in a future column.

The yearly price range for the common stock is shown near the bottom of figure 3. We'll let our spreadsheet program do the division for us. Starting at C39, enter (98+55.625)/2, (71.5+48.375)/2, (72.75+50.375)/2, (80+61.125)/2, and (77.125+58.375)/2. The me-

dian share prices for each year should appear: 76.81, 59.94, 61.56, 70.56, and 67.75.

Now you're finished entering IBM's income statement. Compare your work against the finished template shown in figure 4.

Now we can turn our attention to the retained earnings section of our model, which starts at line 41. In IBM's financial statements, the information we need appears immediately below the "Net Income."

Move your cursor to C43 and enter the retained earnings balance at the beginning of each year: 13909000, 12491000, 11012000, 9575454, and 8677680. Notice that in the next line the net income for each year has automatically been transferred from the income statement. This plus the beginning balance is added into the TOT RET. INCOME line in row 45. Next, move your cursor to C49 and enter the cash dividends for each year: 2053000, 2023000, 2008000, 1505962, and 1763102.

We're going to have to modify our spreadsheet to accommodate the next two items in IBM's statement of retained earnings. These are "Capital stk. purch. & canceled" and "Purch. & sales of treas. stk." There is an extra row under TOTAL DIVIDENDS where we can add this information. Move your cursor to A51 and enter the title STOCK TRANSACTIONS. (In *VisiCalc* you'll probably want to split this label between two cells.) Starting at C51, enter the stock transactions data from IBM's statement of retained earnings: 6000, 4000, 75000, 68714, and 393781+55911.

Now we have to change the formulas in row 52 to include the data in row 51. Move your cursor to C52 and you'll see at the top of the screen that the current formula is +C45−C50. To deduct row 51, type +C45−C50−C51. Now you'll want to replicate this formula to other cells in row 52. In row 52, the correct year-end balance should be: 16259000 for 1982, 13772000 for 1981, 12491000 for 1980, 11012037 for 1979, and 9575454 for 1978.

Astute readers will notice that the beginning balance for 1982 is not equal to the ending balance for 1981. In 1982, IBM changed its accounting practices regarding foreign currency transactions. Currency adjustments are accumulated in a separate component of stockholders' equity, termed the translation adjustment. We won't dwell on this point; we just wanted to show that there was a rational explanation.

Again, compare your work against figure 4.

Now we can move on to the balance sheet—which starts in line 54 of our model. Move your cursor to C58 and start by entering the cash data for each year: 405000, 454000, 2810000, 297839, and 273799. Do the same for marketable securities, starting in C59: 2895000, 1575000, 1831000, 3473384, and 3756887.

We put the values across from "Notes & accounts receivable (net)" in the IBM balance sheet into row 60. Move your cursor to C60 and enter 5433000, 4792000, 4877000, 4671315, 4134458. The next item of interest is "Inventories," which we enter in row 61: 3492000, 2805000, 2293000, 1841791, and 1561382.

In the row marked OTHER CUR. ASSETS, we can put in IBM's prepaid expenses. Starting at C63, enter 789000, 677000, 643000, 566436, and 594385.

In line 64, IBM's total current assets will automatically appear. Check these figures against the totals shown in figure 3. Be sure to do a manual recalculation before checking.

Let's proceed to the fixed assets section of our model. In row 70, PROP., PLANT, EQU., we want to enter the data across from "Plant, rental machines & other prop., at cost" in IBM's balance sheet. Move your cursor to C70 and type 30767000, 30136000, 26370000, 22744188, and 19175171. In the next row we can enter the accumulated depreciation (be sure to include the minus signs): −13204000, −12858000, −11353000, −10551169, and −9872943.

Line 80 holds the last of IBM's assets, listed in the balance sheet as "Deferred charges & other assets." Enter these figures in the row labeled OTHER in the INTANGIBLE ASSETS section of the model. Once you've entered all the assets, check your total (shown in line 84) against the total assets figures shown on the balance sheet.

Moving right along, we come to the liabilities section of our model. Referring again to figure 3, enter the amounts shown for "Accounts payable & accruals" into line 89 of our model, in the section labeled CURR. LIABILITIES. We don't have any notes payable to put into the next row, but notice in the balance sheet the dividends payable in 1978. Let's change the label in A90 so we accommodate this figure. Move your cursor to A90 and type DIVIDEND PAYABLE (again, if you're using *VisiCalc*, split the label over two columns). Now move to G90 and enter the figure 501610.

Starting at C91, enter the amounts shown for "Loans payable" into the model, in the row marked LOANS PAYABLE. The final current liability is listed as "U.S. Fed & non-U.S. inc. taxes." Once you've entered this amount, check to see if your total current liabilities match the figures shown on the IBM balance sheet.

We can now enter the data on IBM's long-term debt. The figures from the balance sheet are entered into row 99 of our model. This is the only entry for this section of the spreadsheet.

Move your cursor down to line 107 in the OTHER LIABILITIES section of the model. Enter the values for "Deferred investment tax credits" in the DEFERRED TAXES row and, below that, "Empl. indem. & retire. plan res." in line 108, which is labeled OTHER LIABILITIES.

This brings us to the final section of our model, the owners' equity in line 120. Our model leaves us room to record two types of preferred and two types of common stocks. IBM has only common stock outstanding. The figures in this line represent the par value per share times the number of shares outstanding. Move your cursor to C122 and type

COMPARATIVE CONSOLIDATED INCOME ACCOUNTS, YEARS ENDING DEC. 31

	(Thousands of dollars)				
	1982	1981	1980	1979	1978
Sales	16,815,000	12,901,000	10,919,000	9,472,649	8,754,794
Rentals & Services	17,549,000	16,169,000	15,294,000	13,390,127	12,321,295
Total sales, rental & services rev.	34,364,000	29,070,000	26,213,000	22,862,776	21,076,089
Cost of sales	6,682,000	5,321,000	4,197,000	3,266,605	2,838,225
Cost of rentals & services	7,006,000	6,695,000	5,952,000	5,146,353	4,645,800
Selling, development & engineering, & general & admin. exp.	12,620,000	11,027,000	10,324,000	9,205,367	8,151,129
Interest charges	454,000	407,000	273,000	140,487	55,175
Miscellaneous other income	328,000	368,000	430,000	449,295	411,808
Inc. bef. inc. taxes	7,930,000	5,988,000	5,897,000	5,553,259	5,797,568
U.S. Fed. & non-U.S. income taxes	3,521,000	2,680,000	2,335,000	2,542,000	2,687,000
Net Income	4,409,000	3,308,000	3,562,000	3,011,259	3,110,568
Retained earnings, beg. of year	13,909,000	12,491,000	11,012,000	9,575,454	8,677,680
Cash dividends (declared)	2,053,000	2,023,000	2,008,000	1,505,962	1,763,102
Capital stk. purch. & canceled					393,781
Purch. & sales of treas. stk.	6,000	4,000	75,000	68,714	55,911
Retained earnings, end of year	16,259,000	13,772,000	12,491,000	11,012,037	9,575,454

COMPARATIVE CONSOLIDATED BALANCE SHEETS AS OF DEC. 31

	(Thousands of dollars)				
	1982	1981	1980	1979	1978
ASSETS					
Cash	405,000	454,000	281,000	297,839	273,799
Marketable Securities	2,895,000	1,575,000	1,831,000	3,473,384	3,756,887
Notes & accounts receivable (net)	5,433,000	4,792,000	4,877,000	4,671,315	4,134,458
Inventories	3,492,000	2,805,000	2,293,000	1,841,791	1,561,382
Prepaid Expenses	789,000	677,000	643,000	566,436	594,385
Total current assets	13,014,000	10,303,000	9,925,000	10,850,765	10,320,911
Deferred charges & other assets	1,964,000	2,005,000	1,761,000	1,486,190	1,148,235
Plant, rental machines & other prop., at cost	30,767,000	30,136,000	26,370,000	22,744,188	19,175,171
Less: Accum. depreciation	13,204,000	12,858,000	11,353,000	10,551,169	9,872,943
Property account (net)	17,563,000	17,278,000	15,017,000	12,193,019	9,302,228
Total	32,541,000	29,586,000	26,703,000	24,529,974	20,771,374
LIABILITIES					
Accounts payable & accruals	4,826,000	4,135,000	3,566,000	3,794,525	3,607,318
Loans payable	529,000	773,000	591,000	932,729	241,484
U.S. Fed. & non-U.S. inc. taxes	2,854,000	2,412,000	2,369,000	1,717,634	1,459,710
Dividend payable					501,610
Total current liabilities	8,209,000	7,320,000	6,526,000	6,444,888	5,810,122
Long term debt	2,851,000	2,669,000	2,099,000	1,589,358	285,534
Deferred investment tax credits	323,000	252,000	182,000	139,349	109,882
Empl. indem. & retire. plan res.	1,198,000	1,184,000	1,443,000	1,395,144	1,072,226
Common stock	5,008,000	4,389,000	3,992,000	3,973,911	3,942,164
Retained earnings	16,259,000	13,772,000	12,491,000	11,012,037	9,575,454
Less: Translatioo adjustment	1,307,000				
Less: Treasury stock, at cost			30,000	24,713	24,008
Total stockholders' equity	19,960,000	18,161,000	16,453,000	14,961,235	13,493,610
Total	32,541,000	29,586,000	26,703,000	24,529,974	20,771,374
OTHER INFORMATION:					
Number of shares (year-end)	602,406,128	592,293,624	583,806,832	583,594,543	583,241,454
Stock price - high	98.000	71.500	72.750	80.000	77.125
Stock price - low	55.625	48.375	50,375	61.125	58.375

Figure 3

FINANCIAL STATEMENT ANALYSIS

INCOME STATEMENT

INCOME ****************	1982	1981	1980	1979	1978
NET SALES	34364000	29070000	26213000	22862776	21076089
OTHER INCOME	328000	368000	430000	449295	411808
TOTAL INCOME	34692000	29438000	26643000	23312071	21487897
EXPENSES ****************					
COST OF GOODS SOLD	13688000	12016000	10149000	8412958	7484025
* GROSS PROFIT	21004000	17422000	16494000	14899113	14003872
GEN. SALES & ADMIN	9578000	11027000	10324000	9205367	8151129
DEPRECIATION					
RESEARCH & DEVEL.	3042000				
TOTAL OPERATING EX	12620000	11027000	10324000	9205367	8151129
* NET OPER. INCOME	8384000	6395000	6170000	5693746	5852743
INTEREST EXPENSE	454000	407000	273000	140487	55175
* INCOME BEF. TAX	7930000	5988000	5897000	5553259	5797568
TAXES	3521000	2680000	2335000	2542000	2687000
* NET BEF MIN & EX	4409000	3308000	3562000	3011259	3110568
MINORITY INTEREST					
EXTRAORDINARY ITEM					
DISCONTINUED ITEM					
* NET INCOME	4409000	3308000	3562000	3011259	3110568
# PREF SH (000S)					
# COM. SH (000S)	602406.1	592293.6	583806.8	583594.1	583241.5
COM. MARKET PRICE	76.81	59.94	61.56	70.56	67.75

****

STATEMENT OF RETAINED EARNINGS

RETAINED INCOME	13909000	12491000	11012000	9575454	8677680
*ANNUAL NET INCOME	4409000	3308000	3562000	3011259	3110568
*TOT RET. INCOME	18318000	15799000	14574000	12586713	11788248
DIVIDENDS PAID:					
PREFERRED					
COMMON	2053000	2023000	2008000	1505962	1763102
* TOTAL DIVIDENDS	2053000	2023000	2008000	1505962	1763102
STOCK TRANSACTIONS	6000	4000	75000	68714	449692
* BALANCE YEAR END	16259000	13772000	12491000	11012037	9575454

****

BALANCE SHEET

CURRENT ASSETS:					
CASH	405000	454000	281000	297839	273799
SECURITIES	2895000	1575000	1831000	3473384	3756887
RECEIVABLES	5433000	4792000	4877000	4671315	4134458
INVENTORY	3492000	2805000	2293000	1841791	1561382
DEFERRED TAXES					
OTHER CUR. ASSETS	789000	677000	643000	566436	594385
TOTAL CURRENT	13014000	10303000	9925000	10850765	10320911
FIXED ASSETS:					
PROP., PLANT, EQU	30767000	30136000	26370000	22744188	19175171
LESS ACC. DEPRECIA	−1.320E7	−1.286E7	−1.135E7	−1.055E7	−9872943
OTHER FIXED ASSETS					
TOTAL FIXED ASSETS	17563000	17278000	15017000	12193019	9302228
INTANGIBLE ASSETS:					
GOODWILL					
OTHER	1964000	2005000	1761000	1486190	1148235
TOTAL INT. ASSETS	1964000	2005000	1761000	1486190	1148235
* TOTAL ASSETS	32541000	29586000	26703000	24529974	20771374
CURR. LIABILITIES:					
ACCOUNTS PAYABLE	4826000	4135000	3566000	3794525	3607318
DIVIDEND PAYABLE					501610
LOANS PAYABLE	529000	773000	591000	932729	241484
INCOME TAX PAYABLE	2854000	2412000	2369000	1717634	1459710
TOTAL CURRENT L.	8209000	7320000	6526000	6444888	5810122
DEBT:					
LONG-TERM DEBT	2851000	2669000	2099000	1589358	285534
OTHER L.T. DEBT					
OTHER L.T. DEBT					
TOTAL L.T. DEBT	2851000	2669000	2099000	1589358	285534
OTHER LIABILITIES					
DEFERRED TAXES	323000	252000	182000	139349	109882
OTHER LIABILITIES	1198000	1184000	1443000	1395144	1072226
OTHER LIABILITIES					
TOTAL OTHER	1521000	1436000	1625000	1534493	1182108
*TOTAL L.T. LIABIL	4372000	4105000	3724000	3123851	1467642
*TOTAL LIABILITIES	12581000	11425000	10250000	9568739	7277764
OWNERS' EQUITY					
PREFERRED					
PREFERRED					
COMMON	5008000	4389000	3992000	3973911	3942164
COMMON					
TRANSLATION ADJ.	−1307000				
RETAINED EARNINGS	16259000	13772000	12491000	11012037	9575454
LESS TREAS. STOCK			−30000	−24713	−24008
TOT. OWNERS' EQUITY	19960000	18161000	16453000	14961235	13493610
*TOT LIA. & OWN EQU	32541000	29586000	26703000	24529974	20771374

****

ANALYSIS SECTION

LIQUIDITY:					
WORKING CAPITAL	1.59	1.41	1.52	1.68	1.78
QUICK ASSET	1.16	1.02	1.17	1.40	1.51
L.T. SOLVENCY:					
DEBT-EQUITY	0.14	0.15	0.13	0.11	0.02
TIMES INT. EARNED	17.47	14.71	21.60	39.53	105.08
ASSET COVERAGE	7.73	7.50	8.69	10.36	47.99
EFFICIENCY					
INVENTORY TURNOVER	3.92	4.28	4.43	4.57	4.79
ACCTS. REC. TURNOV	57.71	60.17	67.91	74.58	71.60
SALES/WORKING CAP.	7.15	9.75	7.71	5.19	4.67
SPECIFIC					
PROFITABILITY					
GROSS PROF. MARGIN	60.17	58.67	61.28	63.20	64.49
NET PROF. MARGIN	12.83	11.38	13.59	13.17	14.76
RETURN ON COMMON	22.09	18.21	21.65	20.13	23.05
EARNINGS PER SHARE	7.32	5.59	6.10	5.16	21.33
PRICE/EARNINGS	10.49	10.73	10.09	13.68	13.58

****

Figure 4.

in the data from the "Common stock" line of the balance sheet.

We can change the next line, CAPITAL SURPLUS, to allow us to enter the "Translation adjustment" from IBM's 1982 balance sheet. Move your cursor to A126 and enter the label TRANSLATION ADJ. (splitting between columns if necessary). Now move your cursor to C126 and enter −1307000.

The final entries are for retained earnings and treasury stock. The retained earnings are entered the same way any other value is entered, but the treasury stock is treated as depreciation. Since the treasury stock reduces the owners' equity, you must enter this number as a negative. Move your cursor to E128 and enter −30000, −24713, and −24008.

You've now entered all the information you need from IBM's statements. Check your work against figure 4.

Now that we've gone to all this effort, let's see what we can learn from the analysis section that begins in line 136.

The analysis section consists of four subsections: liquidity, long-term solvency, efficiency, and profitability. The exact way each ratio is calculated is explained in the November and December installments of this column. The category SPECIFIC, shown as the last of efficiency ratios, has intentionally been left blank. You can use this empty row for any special ratio you like. The completed analysis section for IBM occupies the bottom portion of figure 4.

So what do all these numbers tell us about this Brobdignagian corporation?

What strikes us first is the remarkable stability of the financial ratios over recent years. It is amazing that a company this size performs so consistently year after year.

Let's look at the ratios a little more closely.

The liquidity ratios are within acceptable limits.

The long-term solvency ratios indicate just how conservative IBM's capital structure is. A debt-equity ratio of 1 or less is generally acceptable; IBM's is in the .10 to .15 range. The shareholders are well protected. Times interest earned should be around 4; IBM's is about 15. The bondholders are also well looked after. Asset coverage should be around 2; IBM's is over 7.

The efficiency ratios point to strong management. Note the trend to faster collections in the accounts receivable turnover ratio. In times of tight money IBM was able to react by reducing the time it took to collect its outstanding receivables. It has reduced its collection period from 75 days in 1979 to 57 in 1982. That's outstanding for a company this size.

The conservative investor can revel in the company's profitability ratios. The consistent profit margins and return on common stock indicate that this investment would look good in anyone's retirement fund.

In short, the ratios help quantify what we already knew about the desirability of IBM's stock.

We've used IBM as an example. Now you can apply this spreadsheet model to the analysis of other potential investments. ▲


IBM

Do You Really Need This Much Real Estate?

Winchester Cathedral

by John Dickinson

When you attach a Winchester disk (also known as a hard or fixed disk) to your PC, you get the same sense of immensity that you may have had the first time you saw something huge like Saint Patrick's Cathedral or the Astrodome. After you've learned to live within the PC's on-line storage limitation of 320,000 bytes per floppy disk, suddenly having thirty or more times that amount on a single disk can seem unreal.

Size isn't the Winchester's only sterling feature. There are other reasons that a hard disk might be a good idea for your system—convenience, speed, security, and even cost, to name the major potential benefits. But not everyone agrees about the benefits of a Winchester disk, so let's look at some of the pros and cons of hooking one up to your PC.

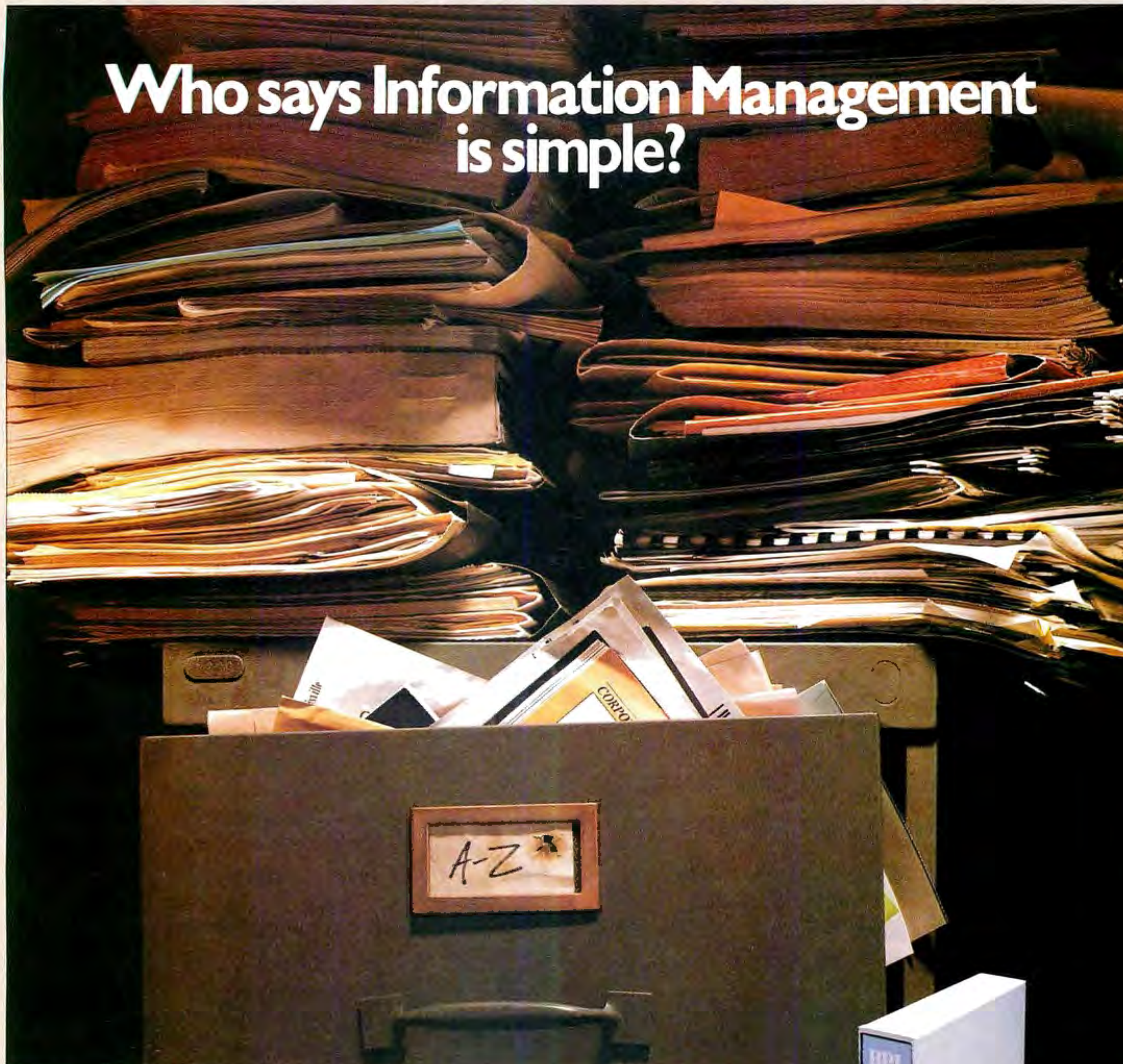
Along with the Winchester's storage capacity comes the convenience of being able to keep most (or all) of your actively used programs and files on it, available to you at a touch of a key. You won't need to spend time searching through a shelfful of floppy disk library boxes trying to find a DOS command or your word processor's files.

And the Winchester's size may mean that you never have to worry about running out of disk space again. Most programs give you little or no protection from overloading a floppy disk and little help if you do overload one. Efforts to recover your data—if you *can* recover your data—can be time-consuming and frustrating. If time and effort mean anything to you, the convenience of a Winchester alone can add a lot to your PC's productivity.

> Depending on your applications, a hard disk may be cheaper for you to use than floppy disks.

On the other hand, Winchester disks can't be carted around as casually as floppy disks. Removable-cartridge Winchesters are being marketed, and these can be carried around in your briefcase, but so far they have proved unreliable. New technologies (including nonfloppy, non-Winchester disks) may soon solve the reliability problem and provide PCs with portable high-capacity disks, but for now you may as well consider a Winchester disk (and the data on it) to be something that stays with your PC.

Who says Information Management is simple?



We do. The BPI Information Management System is a revolutionary solution to the complex problem of information storage, retrieval, and reporting. Like all BPI software, it is amazingly easy to understand and use. But, more importantly, the Information Management System offers so many versatile features that there is no limit to its applications at home or in the office. Designed by BPI and AIMS+PLUS, the system is simply the most flexible and powerful data base management product available today for personal computers.

The system combines data base features with word processing and graphics. This combination of functions makes the Information Management System an ideal solution to any problem requiring large storage capacity, rapid entry, simple editing, recall analysis, comparative graphics, or reporting flexibility.

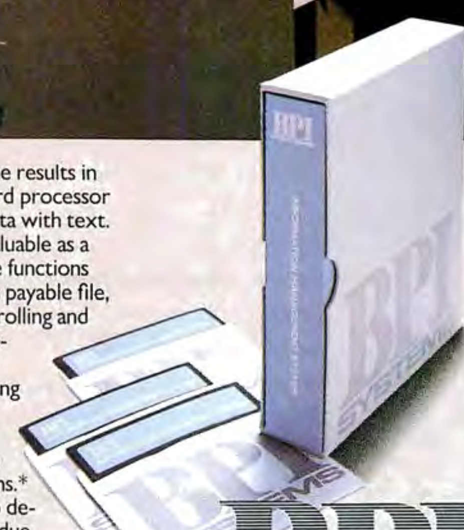
Built-in features allow you to store large amounts of information, search and compare information according to any criteria you desire, and edit information at any time without rebuilding files.

You can design math formulas and employ statistical analyses, then display the results in graph form. A self-contained word processor even enables you to merge file data with text.

In the office, the system is invaluable as a time-saving alternative to routine functions like maintaining an open accounts payable file, updating personnel records, controlling and analyzing fixed assets, creating detailed client information files and even customizing a time accounting system. And, through BPI's new File Transfer Utility Program, information may be retrieved from any of the BPI Accounting Systems.*

At home, the system will keep detailed files on bills and payments, due dates, and remaining balances. It will even store phone numbers and recipes and is a perfect appointment calendar.

Word processing, comparison graphics and easy-to-access, easy-to-edit file storage. BPI and AIMS+PLUS have designed the most versatile and powerful applications generator product on the market. The Information Management System is the simplest solution available to the problem of data base management.



* Those BPI Systems available on MS-DOS™ / A trademark of MicroSoft™.

BPI SYSTEMS™

SIMPLY, THE MOST POWERFUL SOFTWARE IN BUSINESS.

3423 GUADALUPE / AUSTIN, TX 78705 / 512-454-2801

Another Winchester drawback is the fact that you cannot move copy-protected programs to it. Some vendors now deliver their copy-protected software with facilities for moving the programs to a Winchester, usually a limited number of times. But if you cannot move the programs you use most, most of the hard disk's convenience will be lost, since you'll have to continue to run the program from a floppy disk.

Other programs are designed so that they cannot be run from or use data stored on a Winchester disk. This is usually because these programs check your system board switches and are set up to use only the floppy disk drives specified by the settings. DOS 2.0 provides the *assign* command to take care of these programs, but it needs to be used carefully (you can lose control of your system if you're not careful about which drive your system files are on). Other programs do not use standard DOS file-access facilities and cannot use the Winchester regardless of the switch settings. There is no way to fool these programs into using the Winchester.

Some of the speed advantages of a Winchester disk will also be lost if the programs you use most often cannot be run from or use data stored on the Winchester. If a program just loads into memory and stays there until you finish with it, the speed loss isn't great. If it is a heavily overlaid program (pieces of it are swapped between disk and memory), every time a program segment is called from the floppy disk, you will have to wait—just as if there were no Winchester helping you out. Some (but not all) copy-protected programs allow data files to be stored on the Winchester, so some performance improvement over floppy disks will be available in these cases.

One further inconvenience for some users is the fact that adding IBM's Winchester to your PC (by adding the Expansion Cabinet or buying an XT) limits you to using DOS 2.0. The IBM Winchester does not support DOS 1.1, the UCSD p-System, or CP/M-86. However, other vendors do support alternative operating systems for their Winchesters, and some operating system vendors offer support for their systems on the IBM Winchester disk.

Winchester Speed

Doing just about anything on a PC equipped with a Winchester disk is faster than doing the same thing on a floppy-bound machine. Access and load times are at least four times faster on a Winchester than on a floppy disk, and this will significantly improve the running speed of all your programs—from your spreadsheet or word processor to booting the system.

The speed of a Winchester doesn't quite match that of a RAM disk (an electronic disk emulator), but it's a good deal less trouble to use than a RAM disk—and lots more secure. You don't have to copy your files to a Winchester every time you want to use them, and you don't have to worry about losing your data if the power goes off. Many people feel, however, that the superior speed and lower cost of disk emulators make them a better choice for enhancing PC performance than a Winchester disk.

In many cases the effective speed of a Winchester is not what the numbers would suggest. For example, when you use an application program requiring frequent disk accesses for data, both the program and data may be stored on the Winchester, but they may be stored physically far apart. When the program needs to retrieve data from the file as well as from a program segment, the disk head mechanism can spend a lot of time moving back and forth between distant physical locations as it seeks the files, slowing the Winchester down considerably. This frequently happens when large database and word processing programs are run on a Winchester disk.

One very nice way to resolve this conflict is to have enough memory on your Winchester-equipped PC to use a RAM disk too. Storing the overlaid program on the RAM disk and the data files it uses on the Winchester (or vice versa—a little experimentation will tell you which method you prefer) can make the seek-time problem disappear. The combination of both kinds of storage can give you the sort of performance that will make you believe you're driving a 3083 mainframe instead of a PC.

Winchester Security

The strength and security of floppy disks is amazing—they have been known to come through some awful situations with data intact. But floppies can easily be damaged by such common menaces as ambient air pollution and dust, not to mention spilled coffee or a pastrami sandwich. And it is quite easy for someone to walk off with (steal) or just plain lose a floppy disk. Because Winchester disks are enclosed in a sealed housing that protects them from the ravages of the environment,

and because they aren't very transportable, they are less likely to be stolen, lost or damaged.

On the other hand, there are no simple and inexpensive ways to back up the ten or more megabytes of data stored on a Winchester disk. The DOS *diskcopy* and *copy* commands provide easy and effective backup facilities for floppy disks, but there is nothing of equivalent convenience for Winchester disks.

DOS 2.0's *backup* and *restore* commands provide a reliable, but somewhat awkward, backup facility for Winchesters. These commands move an entire Winchester disk, a directory, a directory and its subdirectories, or specified files between the Winchester and a series of floppy disks. (It takes more than thirty disks to hold a full ten-megabyte Winchester.) The file-specification options for *backup/restore* include specifying only those files modified since the last backup or a particular date.

But the *backup* and *restore* commands are relatively slow and clumsy because they require frequent changing of floppy disks and because the data must be restored to the same directory on the Winchester from which the files were backed up. The files cannot be moved from the backup floppy to a normal DOS-format floppy, nor can the backup disks be used as ordinary floppy disks. This can be a problem if your Winchester is down and your only alternative is a floppy-bound machine. (This is by the book, but in fact it's possible to use ASCII text files after some careful editing of a header record placed in them by the backup command. It's likely that your executable and non-ASCII data files can also be used after some similar surgery.)

The regular DOS *copy* command will work as a sort of manual alternative to the *backup* and *restore* commands. Backup floppy disks made with the *copy* command can be used as you would use ordinary DOS disks, and this can be a big advantage. But some users feel that this method is not sophisticated enough. The *copy* command is limited to the capacity of a single floppy disk and, unlike the *backup* and *restore* commands, which have automatic file-selection facilities, it forces you to decide exactly which files to back up.

You can get cartridge tape devices, called tape streamers, that back up or restore your entire Winchester very quickly (usually in less than ten minutes). But they're expensive, costing about half the original price of the disk. A tape can cost nearly as much the other, more obvious alternative—a second Winchester to back up the first one. Of course, a removable-cartridge Winchester would be an attractive backup alternative if the technology were more reliable.

Another problem with Winchester disks is their vulnerability to head crashes. Winchester disk drives are much more delicate than their floppy cousins. The head that reads and writes your data flies over the disk's surface at a very low altitude, while a floppy head drive sort of scrapes along the surface like a broom. Like any other airborne device, a Winchester's head can crash-land, destroying your data in the process. This disaster may be caused by something as innocent as a person's bumping into the PC or XT or as sinister as a power blackout. As Winchester technology continues to develop, such crashes become more rare. But they will always be possible. Careful backup practices are your only protection against a head crash—or any other repair problem that might temporarily or permanently disable your disk and data.

Storage capacity is just about the only indisputable advantage of Winchester disks over floppy disks.

One last security problem is the lack of password protection for data and programs stored on a PC's hard disk. This really has more to do with DOS file-management facilities than with Winchesters in general, but the problem is much more significant on hard disks than it is on floppies.

If you want to protect data stored on floppy disks against theft or improper access, just store the floppies in a safe or locked area. Hauling a Winchester off to the office safe every night is hardly convenient, however, and if you need to protect files from access by other users of the disk, storage in the safe does no good at all. If you need to protect your Winchester files you must provide your own password protection scheme. A locking cabinet and key switch for your PC (and its Winchester if it's in a separate box) aren't bad investments either.

Winchester Cost

Adding a Winchester disk costs roughly half what your PC originally set you back, and, when only the initial cost is considered, owning one appears to be quite expensive. For the first-time or small-scale user it may even make the cost of owning a PC prohibitive. As a result, most vendors, including IBM, offer a Winchester as an optional disk configuration that can be added at a later time. Or a PC owner can buy or trade up to the higher-priced XT, which has a Winchester already installed. If you want to wait awhile, you can save money; Winchester prices will almost certainly come down.

Depending on your applications, a hard disk may be cheaper for you to use than floppy disks. If you have the sort of large-scale application that runs unproductively or can't even be implemented on floppy disks, then cost may not be much of an issue. If you have many applications with small data-and program-size requirements, then it is possible that if you organize yourself and your applications well, a Winchester will be less expensive—regardless of its hardware cost.

When you have a Winchester, there is little temptation to start up a new floppy disk (and probably a backup for it) every time you begin a new project (which may use only a small percentage of the floppy's capacity). There is also less of a tendency to keep a myriad of unnecessary (and possibly out-of-date or confusing) backup floppies around. For example, if your working copy of a program is stored on a Winchester, only one original or original backup of it is necessary, since your working copy will rarely, if ever, need replacement. With the average cost of floppies at about five dollars for a double-sided disk, this can mean a big savings as your applications storage requirements grow.

Winchester Use

Storage capacity is just about the only indisputable advantage of Winchester disks over floppy disks. The IBM Winchester that comes with the XT or the Expansion Cabinet holds ten megabytes of data and programs (that's 10,592,256 bytes). Other vendors offer PC-compatible Winchesters capable of holding up to thirty megabytes, and you can bet on a horsepower race for Winchester size in the future.

Floppy disk storage on the PC is limited to 320K (DOS 1.1), 360K (DOS 2.0), or 400K (UCSD p-System) unless you opt for quad-density drives or nonstandard formats. To give you an idea of floppy disk size, any of these formats will hold well over two hundred pages of written text. That's quite a lot of data but nothing like the amount a Winchester can hold.

Now you should ask yourself if you really need all of a Winchester disk's real estate. There are two ways to think about it. If you're a user with a large database requirement, it's almost a foregone conclusion—personnel or in-

ventory applications, which need large files, can easily use up a significant percentage of a Winchester's space. Only a few application packages and languages are capable of creating and using a file that is spread across several floppy disks, so the size advantage of a Winchester should be obvious. In fact, some large database applications are now being marketed with a Winchester bundled into the package.

Winchesters offer other benefits as well for the user of large databases. Effective use of large databases requires on-line file storage with almost immediate access to the data; implementations requiring frequent changes of floppy disks would hardly meet this requirement. The convenience and speed a Winchester offers these users are unbeatable.

Cost is probably not a factor for large database users. The development or purchase cost of major applications usually far exceeds the cost of a Winchester disk. These same costs make even the most expensive of backup mechanisms economical. Protecting a major database from any sort of damage is of critical importance, so a tape streamer or second Winchester disk is almost imperative for adequate backup—the DOS *backup/restore* facility is probably just not up to the job for these applications.

Security is almost certainly an issue for users of large databases. A reliable cartridge Winchester (if one existed) could be removed and stored in a locked area, just like a floppy disk. It would also make a convenient off-site backup device, as would a tape streamer. At a minimum, a Winchester-based database application should have system-and data-access passwords; a keyswitch-protected and locked system also wouldn't hurt.

If you are a user, or group of users, with many smaller application packages, programs, and files, the question of whether to buy a Winchester disk is more difficult to answer.

One way to find out is to go through your floppy disk library and count the disks you actively use. Don't include backups in your count, but do count everything else, including the office bowling league program, the Christmas card list, and your record collection. If you have more than thirty or forty active floppies, you're probably a good candidate for a Winchester disk.

The size and convenience of the Winchester's on-line storage alone can make your life easier and more productive. Switching between applications stored on a Winchester is easy and quick. And, unless you're a gypsy, the lack of transportability can be overcome by keeping mobile files on floppy disks, or by just copying files to floppies when the need to move them arises.

Security and backup issues can be important, especially if a group of users share the system. In such cases, individual users should

be made responsible for backing up their own files, but a system administrator should be appointed to organize and maintain the system, including maintaining adequate backups of central files and programs.

The difficulty for these users is in deciding how to organize and use all the space provided on the bigger disks. Many PC users have a difficult time keeping themselves organized on floppy disks and shrink at the thought of a Winchester's virtually unlimited space and lack of restriction on the number of files that can be stored on it. DOS 2.0's hierarchical directory facility provides a satisfactory solution, but it may be difficult to use the first time you try it.

When the hierarchical directories are being used, each application can be assigned its own directory, and all the files pertaining to the application can be stored in that directory regardless (almost) of size. A directory's file storage is logical, not physical—there is no size limit to any directory that you define, and files are not stored together physically.

Directories also contain subdirectories, which may contain smaller pieces of the application that tend to fit together. If several users share the Winchester, each can be assigned his or her own directory, which can be further divided into individual application areas.

DOS provides two key facilities for using the directories: current directory and path.

The current directory (set with the *chdir* or *CD* command) specifies which directory is currently active. Unless you specify otherwise, the DOS commands you enter are carried out on the current directory (most DOS commands give you the option of specifying directories other than the current directory if you wish). The whole idea of current directories is an extension of the DOS default drive concept and can be used on floppy disks as well as Winchesters. However, it was clearly designed for use on the big disks.

The path facility provides a default override of the current directory for programs you request. If you try to run a program that isn't stored in the current directory, DOS will search the directories and other disks (including floppies and RAM disks) that you have specified in your path (by means of the *path* command). The order of search is the one you enter when you issue the *path* command. The path facility can be quite useful in a multiple-user situation. Central programs, such as DOS disk-resident commands or the editor that everyone uses, can be stored in a centrally available directory that is always on the users' paths.

However, there are serious flaws in both the directory and path facilities that can affect both single- and multiple-user situations. A major problem of the path facility is that it can be searched only for programs, not for data files. This can cost you quite a bit of space if several directories require standard control files for a centrally stored program normally available on a path.

Worse, if a program is overlaid (and this applies even to compiled programs that use a run-time library), the overlay segments are considered by DOS to be data and must be kept in the current directory. If several users are sharing the Winchester disk, this problem becomes even more serious as multiple copies of files get scattered across the disk. This restriction on path function seriously affects the usability of subdirectories.

The biggest difficulty with the directory facility is that many applications have not been upgraded to take advantage of it. This includes many of IBM's own products, such as *Personal Editor* and some of their compilers.

Personal Editor does not recognize names of files that are not in the current directory, and it insists that its profile and help files be in the current directory. Compilers cannot compile programs stored in another directory, and their file-management routines won't allow you to specify more than the filename and disk drive. Many software authors are concerned that an upgrade could cost their products DOS 1.1 compatibility, requiring them to maintain two different versions of their programs. And many of them use IBM compilers that do not have access to the directory facilities.

Also missing from DOS 2.0 is the ability to override the current directory if you simply want to run a program that is not in it and not on the current path. To use a program in another directory, you must either change the current directory or the path.

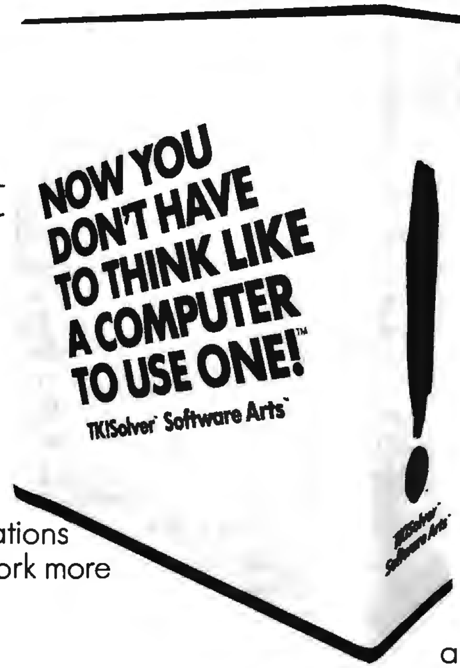
Future versions of DOS may well correct these problems. Many users feel that DOS 2.0 was brought out in a hurry and never finished or thoroughly debugged. If, as some rumors have it, IBM is heading for a multiuser operating system for the PC, the restrictions of the path facility could be fatal to smooth operation of such a system.

Winchester— Worth It?

Should you invest in a Winchester for your PC? As you've seen, there are certainly some problems with having a hard disk, most of them due to deficiencies in DOS 2.0. But adding a Winchester disk to your PC is a lot like adding an air conditioner to your car. The first time you use it there are adjustments to be made and new things to learn about—things that may not seem comfortable at first but that you have to cope with to use the new addition. Once you're accustomed to it, though, you'll never want to go back to being without one! ▲

TK!SOLVER™

THE WORLD'S FIRST EQUATION PROCESSOR FOR PERSONAL COMPUTERS™



The TK!Solver™ program is a professional tool that allows you to work easily with equations. Whether your problem is a simple formula or a model consisting of many equations, TK!Solver can help improve your productivity. Once the equations are written, enter the known values, press the ! key, and TK!Solver gives you the answer.

Engineers, scientists, architects, financial analysts and planners, educators, researchers, and other professionals who use equations and mathematical models can work more creatively with TK!Solver.

TK!Solver provides you with:

BACKSOLVING

If the programs you use now require you to rewrite the same equation to solve for different unknowns, TK!Solver can dramatically improve your productivity. Enter your problem once and then solve for the unknowns no matter where they are in your equation.

ITERATIVE SOLVING

If TK!Solver can't solve an equation directly, take an educated guess at the answer. Type the ! key and the TK!Solver program starts with your guess and performs repeated approximations to converge on the answer.

LIST SOLVING

Given a list of input values, TK!Solver automatically calculates the equation for every value in your list. For example, if you want to know how different interest rates will affect monthly loan payments, enter a list of interest rates and let TK!Solver calculate the payment amount for each value.

UNIT CONVERSIONS

Any type of unit conversion – Fahrenheit to Celsius, meters to feet, dollars to deutschmarks, newtons to dynes – can be made without altering your equations. Just define the numerical relationship between two units of measurement and the TK!Solver program automatically converts the variable value to the unit you specify.

TABLES AND PLOTS

Quickly generate tables and plots of your results on your screen or printer.

If you can define a problem mathematically, the TK!Solver program can help you find the answer. It's a flexible problem-solving tool that you can apply easily to a variety of mathematical problems.

AVAILABLE NOW

You can run the TK!Solver program on the IBM® PC and XT and compatible machines, the Digital™ Professional™ 350, the Digital™ Rainbow™ 100, the Wang Professional Computer, Apple® //e, and on the following personal computers using MS™-DOS: TI Professional Computer, GRiD Compass Computer,™ Canon AS-100, Eagle® 1600, Toshiba T300, and the Zenith Z-100.™

Software Arts™

The creators of VisiCalc® and TK!Solver™

marketalk news

Unless otherwise indicated, software listed runs in DOS on machines with either display adapter and requires 64K and at least one disk drive.

Δ Two software enhancements to the *Condor* database have been announced by **SoftHelp** (4237 Cole Avenue, Dallas, TX 75205; 214-559-3095). *Help! Condor* provides on-line help screens for the database and displays their use. $75. Δ *Condor Menu* makes the database menu-driven, eliminating the necessity of learning commands or syntax. $125. Δ *Condor 3 User's Guide with Applications* teaches you tricks not found in the manual. $25.

Δ A set of packages that provide law firms and corporate legal departments with legal management has been released by **Master Data Center** (24700 Northwestern Highway, Southfield, MI 48075; 313-352-5810). The menu-driven *Master Legal System/PC* contains a module for general legal matters as well as modules for patent and trademark records. Features include multiple security levels, automatic indexes, a common database, and user customization of files. Modules work as standalones or can be used together. First module: $5,000. Second: $3,000. Third: $2,500.

Δ First-time PC users can teach themselves how to use their computer and get an overview of practical applications using *PC Master* from **Courseware** (10075 Carroll Canyon Road, San Diego, CA 92131; 619-578-1700). The three-disk package includes an applications demonstration disk that, for example, shows how to create, edit, and print a document. XT version incorporates the latest features of DOS 2.0. $79.95.

Δ Create up to four disk emulators with 32K to 360K capacity each with the *Insta-Drive 3.02* utility from **Persyst** (15801 Rockfield Boulevard, Irvine, CA 92714; 714-859-8871). The program provides instant access to data by allowing frequently used programs and files to reside in main memory. In addition, a physical disk drive can be replaced logically with a RAM-emulated disk. *Waitless Printing 3.02* is a buffer program allowing a user to continue working while a printer is being run. Requires printer adapter or RS-232 interface; works with any industry standard dot-matrix, daisy-wheel, thimble, or band printer. Both for $100.

Δ *1040Plan* is a template for income tax planning and preparation that's compatible with *1-2-3*, from **1040Plan** (1125 Sunnyhills Road, Oakland, CA 94610; 415-451-7090). The package can compute tax forms for small businesses as well as individual users. Requires *1-2-3* and 256K. $45.

Δ A program that adds a novel touch to birthdays and other special events, *Personal Message* prints a message in seven-inch-high characters onto standard fan-fold paper. Select from a menu of messages or create your own. From **ATC Software** (Route 2, Box 448, Estill Springs, TN 37330; 615-967-9159). $19.95.

Δ *Astrocalc* is astronomical software for the amateur or professional astronomer. From **Zephyr Services** (306 South Homewood Avenue, Pittsburgh, PA 15208; 412-247-5915). The program calculates and displays all basic data for the sun, moon, and all planets based on date, time, and location information. Rising and setting times, right ascension and declination, ecliptic latitude and longitude, and Julian day number are only a few of the aspects calculated. $29.95.

Δ The sponsors of the *West Coast Computer Faire* are calling for papers to be submitted. The ninth annual event will be held March 23 through 25, 1984, in San Francisco. Parties also interested in submitting papers for the second annual *PC Fair*, to be held October 26 through 28 in the same city, should contact **Computer Faire** (570 Price Avenue, Redwood City, CA 94063; 415-364-4294).

Δ *Softcon*, the international conference and trade fair for the software industry, will be held February 21 through 23, 1984, at the Superdome in New Orleans. Sponsored by **Northeast Expositions** (822 Boylston Street, Chestnut Hill, MA 02167; 800-841-7000). Registration is $30 for exhibits only and $195 for the full conference program.

Δ **Counterpoint Software** (4005 West Sixty-fifth Street, Minneapolis, MN 55435; 800-328-1223) has released three new games in their educational Early Games Series. *Matchmaker* is a reading-readiness program for preschoolers that playfully guides them through a series of matching games involving colors, lines, and shapes. $29.95. Δ *Piece of Cake* is designed to teach elementary math in five games that take place in a magical bakery. Children keep count of cakes as they come out of the oven, learning to compute quickly, before the cakes fall. $29.95. Δ *Fraction Factory* teaches elementary and junior high fraction concepts with animated hints and playful sound. $29.95. Early Games series requires color/graphics adapter. Δ *Quizagon* is an adult and family game that challenges players with more than six thousand questions in such categories as history, sports, words, and entertainment. Categories in development: comics, old movies, and rock 'n' roll. Two disks. $39.95.

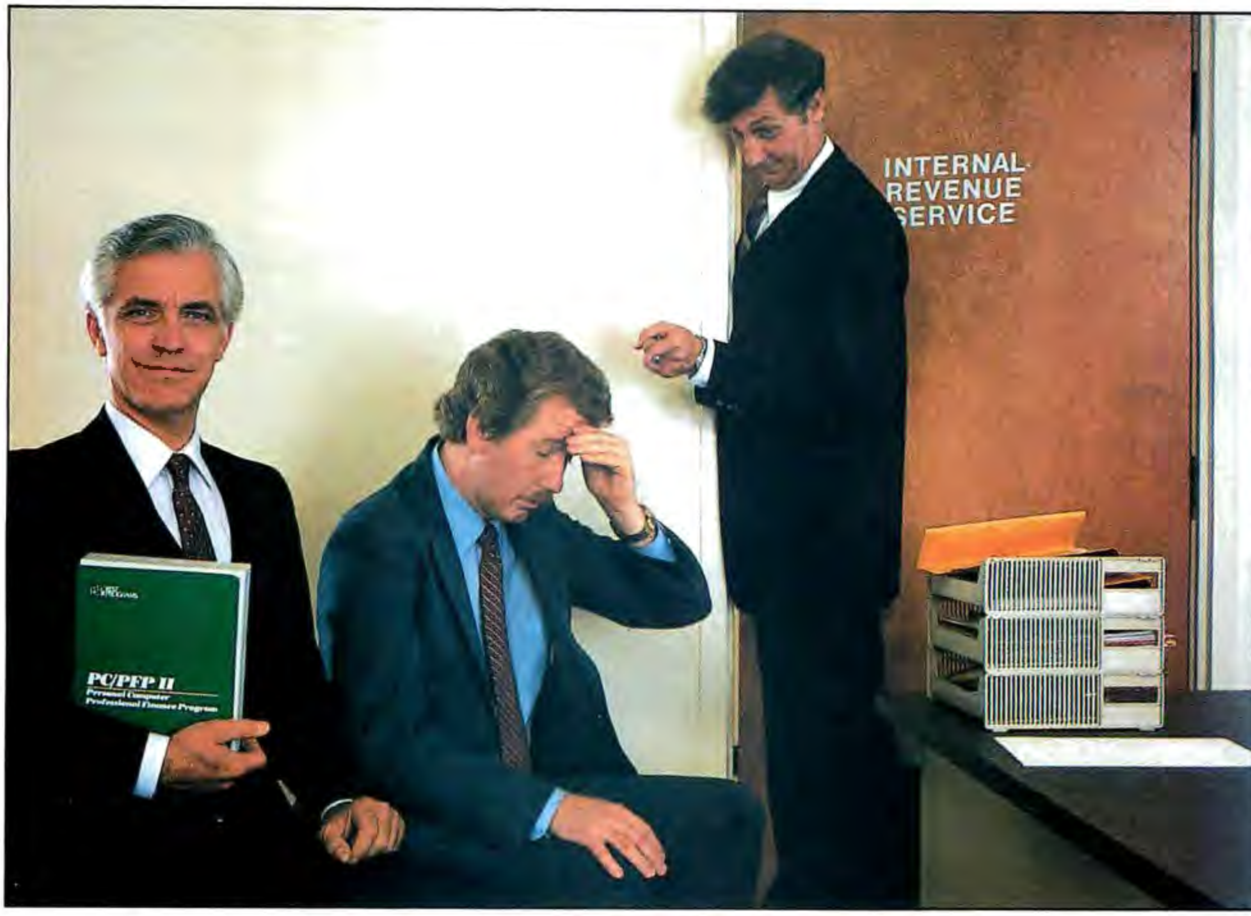
Δ *The Sorting System Part I* explains and demonstrates ten important sorting algorithms using color or monochrome graphics. Programs may be run with your data. From **General Computing** (145 Summit Drive, Cedar Falls, IA 50613; 319-277-7105). The Basic source code for and description of algorithms is included. $45.

Δ IBM PC monitors can now be provided with the capability of instantaneous response to on-screen touch commands with *Soft-Touch* from **Computer Technology Associates** (1704 Moon N.E., Albuquerque, NM 87112; 505-298-2140). The bezel and interface uses infrared emitter-sensor array technology that allows moving stimuli to be tracked. Can be accessed directly through Basic without extra assembly language routines. Fits the IBM or Amdek Color II monitor. Hardware and software included. $695.

Δ **Ring King Visibles** (2210 Second Avenue, Muscatine, IA 52761; 319-263-8144) has introduced the *MDT070*, a locking storage box for 5-1/4-inch disks. The molded, smoke-gray plastic container features two built-in carrying handles, hinged lid, and interior dividers. Holds up to seventy disks. $24.95.

Δ The multimode *DS220* matrix printer from **Datasouth** (4216 Stuart Andrew Boulevard, Charlotte, NC 28210; 704-523-8500) provides correspondence printing for word processing, draft-quality printing for high-speed data processing, and dot-addressable graphics. Four standard fonts are built-in, as well as variable pitch draft fonts and a special micro-character set. Seven international character sets are also resident. $1,995.

Δ For use in language instruction or actual programming, a version of the Pascal programming language has been released by **New Classics Software** (239 Fox Hill Road, Denville, NJ 07834; 201-625-8838). *Pascal 80* is a version of Standard Pascal and writes ASCII source, p-code, and .com files, with compilation from memory or disk and an include



Best's Professional Finance Program– Financial Aspirin for Tax Time Headaches.

It's hard to find the time to keep track of personal or small business finances, or to get the training needed to produce sophisticated financial reports. The result is often a makeshift accounting system that can cause a lot of headaches every April 15. But now Best Program's Professional Finance Program (PFP II) makes it easy for you to have all of the complex accounting tools you need right at your fingertips.

PFP II is a powerful record-keeping system that helps you organize 26 asset accounts like checkbooks, IRA's and money markets, as well as 26 liability accounts like credit cards, loans, and mortgages. The system also sets up over 1,000 income and expense categories, 99 sort categories for taxes and expenses, an electronic address book, and automatic check printing and reconciliation. In short, PFP II keeps track of all of your key financial records.

However, the Professional Finance Program is more than just a record-keeping system. From your input it provides eight basic types of financial reports. You can modify these to produce dozens of custom-tailored reports on things like income and expense summaries, budgets, and net worth. PFP II also tracks all asset and liability account activities and provides bar graphs for cash flow, income, and expense reports. Best's Professional Finance Program makes accounting a lot easier, allowing you time to *manage* your finances instead of *organizing* them.

And don't worry if you don't have any computer or accounting skills. The user's manual includes a tutorial that guides you through the use of the program in six easy lessons.

PFP II runs on the IBM Personal Computer (and the model XT) with a minimum of 128KB memory, and two diskette drives, one of which must be double sided.

PFP II is fully integrated with PC/TaxCut™, a tax planning and preparation program. Simply take the information provided by Best's Professional Finance Program and feed it into PC/TaxCut for an automatic printout of various tax forms. PC/TaxCut and PFP II provide you with a complete tax and financial package that can save you the worry often associated with financial and tax management.

You can order your PFP II for $245 and PC/TaxCut for $255 by using Visa, Master Charge or American Express. To order call toll-free, **1-800-368-2405**. In Virginia call 703-931-1300. Or write to Best Programs, 5134 Leesburg Pike, Alexandria, VA 22302.

BEST PROGRAMS

"The Quality Software Company"

function to allow reuse of library procedures. $79.

Δ **Personal Systems Publications** (Box 90754, Los Angeles, CA 90009) has released a two-hundred-page repair manual called *How To Repair and Maintain Your Own IBM PC/XT.* With step-by-step instructions, the book details the most probable hardware problems and offers easy trouble-shooting tips. Written by a team of experienced technical and engineering writers. $19.95.

Δ Two handy guides from **Alfred Publishing** (15335 Morrison Street, Sherman Oaks, CA 91413): *The Personal Computer Glossary* is a sixty-four-page glossary of computer terms with definitions. $2.95. Δ *How To Use the IBM Personal Computer,* by George Ledin, Jr., contains sixty-four pages of basic instructions on setting up, expansion, applications, and maintenance of your PC. Includes sections on the XT and DOS 2.0. $2.95.

Δ *DOSshell* for the XT creates an organized filing system on the hard disk by setting up predefined directories and subdirectories. From **XTC Handcrafted Software** (2019-C Bath Street, Santa Barbara, CA 93105; 805-687-0865). The software contains menus and help screens. $70.

Δ The *Simultaneous Sample & Hold* data-acquisition system is a multifunction board that provides simultaneous analog input and analog output plus digital I/O and clock functions. From **Data Translation** (100 Locke Drive, Marlboro, MA 01752; 617-481-3700). The board combines the utility of multichannel signal freezing with the flexibility and low cost of personal computer-controlled data acquisition. Manual includes example Basic programs. $1,585.

Δ The *IBM XT/PC 68000 Workstation* is a thirty-two-bit color graphics system for engineers, architects, and designers that has been introduced by **Summit CAD** (5222 FM 1960 West 102, Houston, TX 77069; 713-440-1468). With the XT as base micro, the configuration comes with all the necessary hardware and software for the creation of two- and three-dimensional graphics for both plotting and processing. Display processing time equals that of many mainframe and minicomputer graphics systems. $7,989.

Δ *Financial Horizons* is a menu-driven program that analyzes and illustrates available alternatives to terminating or retiring participants of profit-sharing or pension plans. From **Horizons Computer** (117 North First, Ann Arbor, MI 48104; 313-761-3155). Output includes tax form data, comparisons of IRA rollover versus ten-year averaging, and a detailed comparison of various investment alternatives. Requires double-sided disk. $550. Demo: $50.

Δ The Associated Press is now providing *P Stock and Videotex* news services on The Source, the on-line information service of **Source Telecomputing** (1616 Anderson Road, MacLean, VA 22102; 703-734-7500). More than two hundred fifty daily dispatches of international, national, sports, business, and weather news is provided, as well as regularly updated stock and bond reports, Dow Jones averages, closing tables of mutual funds, and other stock and commodities information. Registration: $100. $20.75 per hour, weekdays. $7.75 per hour, evenings and weekends.

Δ Along with IBM's announcement of the PC*jr* comes news of software packages, both cassette and disk, specially designed to run on the new home machine. Some programs require DOS 2.1, an enhanced version of DOS 2.0. The following games from **IBM** (Box 1328, Boca Raton, FL 33432; 800-447-4700) are available on cartridge: *Mouser,* a fast-paced, arcade-style game where the player is a farmer in a nine-room farmhouse overrun by mice with the object of the game to trap all the mice in the rooms by using movable walls, $35; *ScubaVenture,* for one or two players, with each player controlling a diver searching a dangerous undersea cavern for sunken treasure, $35; *Crossfire,* where a player must defend the gridlike streets of a city from a swarm of aliens, $35; and *Mine Shaft,* where a player maneuvers a mining car around dangerous mine shafts looking for a fortune in diamonds, $35.

Δ Disk software for the Junior, also from IBM, includes *Animation Creation,* a creative entertainment program allowing the user to create animation sequences, draw color pictures, copy them, make small changes, and display them on a color display. $40. An entertaining and educational program, *Monster Math* gives a student the opportunity to add, subtract, multiply, and divide on six different levels. Problems can be mixed, and points are scored for correct answers. Requires PC*jr* Basic cartridge. Also available for the PC. $30. *Adventures in Math* is a game that allows young people to improve their basic math skills as they make their way through three types of castles. Treasures found along the way are rewards for problems correctly solved. Requires PC*jr* Basic cartridge. Also available for the PC. $45. *Turtle Power* introduces the Logo programming language to children. The package is a "drawing board" that provides immediate feedback without preliminary instruction. $50. A reduced version of the PC package, *Home Budget,jr* is designed to help structure a household budget by allocating income to categories of spending. Requires PC*jr* Basic cartridge. $45. *Personal Communications Manager* enables the Junior to send and receive correspondence with compatible computers over standard phone lines. Requires double-sided drive, internal modem, or asynchronous communications adapter with a supported autodial modem. $100.

Δ *Bumble Games* is a set of six game programs for children, introducing the use of number pairs to name positions in an array or points on a grid. Children are guided by Bumble, an imaginary creature from another world. From **The Learning Company** (4370 Alpine Road, Portola Valley, CA 94025; 415-851-3160). $40. *Bumble Plot,* also featuring the Bumble character, is a set of five programs building on the graphic skills introduced in *Bumble Games*. The games are arranged according to the level of difficulty. $40. *Juggles' Butterfly* is a set of three game programs introducing children to a computer before they can read. It divides the keyboard into quarters with a template, and children press keys to make pictures appear. Simple directional concepts, circles, lines, and

some alphabet letters are explored. $35. All three packages require PC*jr* Basic cartridge and are also available for the PC.

Δ *HomeWord* is an easy-to-learn word processor from **Sierra On-Line** (Sierra On-Line Building, Coarsegold, CA 93614; 209-683-6858). Includes Lisa-like picture menus, color, and a custom keyboard overlay. Can be used with a television (with the IBM RF Modulator) or any monitor. $75.

Δ Some earlier PC programs from IBM that also run on the Junior are *EasyWriter, PFS:File, PFS:Report, Dow Jones Reporter, Adventure, Adventure in Serenia, Basic Compiler* (must have sufficient storage for compiler and link), *Casino Games, FileCommand, VisiCalc 1.20,* Logo, *Multiplan, Strategy Games, Time Manager, Word Proof, Macro Assembler, Personal Communications Manager, Personal Editor,* and *Professional Editor.* DOS 2.1 and PC*jr* Basic cassette requirements vary for each package. Some programs require an eighty-column monitor.

Δ Publications from IBM specifically for the PC*jr*: *Hands-on Basic for the IBM PCjr* is an illustrated, easy-to-follow manual intended to teach fundamental concepts and explain all the features of built-in Basic, as well as the cartridge Basic extras. $17.50. *Basic Made Easy for the PCjr* is an introductory manual for beginners, taking them through the more commonly used commands, statements, and functions of the language. $13. *IBM PCjr Technical Manual* provides hardware design and interface information. Intended for programmers, engineers, and designers. $35. *IBM Personal Computer Disk Operating System Technical Reference* contains detailed technical information not contained in the manuals available with DOS 2.1. $30. *IBM PCjr Hardware Maintenance and Service Manual* is used to isolate and repair any failure of the field-replaceable unit. Includes an explanation of the power-on diagnostics. $88.

Δ *Personal Accounting, Speed Reading,* and *Personal Investing* are three home packages for the Junior from **BPI Systems** (3423 Guadalupe, Austin, TX 78705; 512-454-7191). The accounting system is an advanced money management tool that lets a user plan a household budget. The speed-reading program is a self-paced course for improving reading skills that can teach at speeds up to a thousand words per minute. The investing package is a securities portfolio analysis and management system for the private investor. Communications software included. Requires modem. $195 each.

Δ **Ampersand** (128 South George Street, Box M-84, York, PA 17405; 717-845-5602) has announced the *Branchbanker/jr* family of financial services marketing programs. More than sixty-five standalone financial marketing and customer service programs can be tailored into an applications disk. Application areas include certificate of deposit sales, mortgage lending, and personal finance analysis. Banks using the *Branchbanker Sellstation* on the XT can tailor their own disks.

Δ The *Norton Utilities*, from **Peter Norton Computing** (2210 Wilshire Boulevard, Santa Monica, CA 90403; 213-399-3948) are now compatible with the PC*jr*. $80.

Δ The speed, comfort, and reduced error rate of the Dvorak layout are offered in a plug-compatible keyboard for the PC from **Key Tronic** (Box 14687, Spokane, WA 99214; 509-928-8000). The home row of the Dvorak keyboard can configure 3,000 word combinations as compared with Qwerty's 100. Familiar placement of shift and return keys and a convenient enter key near the numeric keypad are offered on the KB-5150D Dvorak board. $269.

Δ *Prisoner 2* is an interactive graphic adventure that pits the player's ingenuity against clever and diabolical captors. Escape from an island requires solving puzzles, overcoming obstacles, and answering riddles. From **EduWare Services** (28035 Dorothy Drive, Box 22222, Agoura Hills, CA 91301; 213-706-0661). $39.95.

Δ June 26 through 28, 1984, are the dates for PCExpo, to be held at the New York Coliseum in New York City. The second annual trade show is sponsored by **PCExpo** (333 Sylvan Avenue, Englewood Cliffs, NJ 07632; 201-569-8542); the show will be repeated in Anaheim, California, September 24 through 26.

Δ Written by computer expert and author Phillip I. Good, *A Critic's Guide to Software for the IBM-PC* offers appraisals of the most popular spreadsheet, word processing, data management, and graphics programs. Published by **Chilton Books** (Radnor, PA 19089; 800-345-1214). Softback. $12.95.

Δ Link the PC to the Unix operating system running on a DEC mini with *INtext II Emulator* software from **Interactive Systems** (1212 Seventh Street, Santa Monica, CA 90401; 213-450-8363). With the emulator, the PC can also function as a VT100 or VT52, as well as a standard dumb terminal. Requires communications adapter. $495.

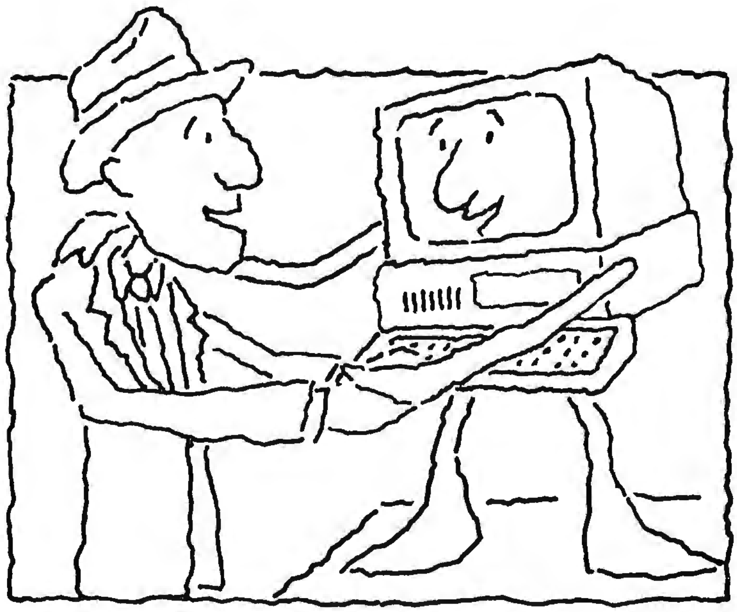
Δ Color and plotter support have been added to *PC-Draw 1.3*, the drawing system from **Micrografx** (1701 North Greenville, Richardson, TX 75081; 214-343-4338). Enhancements also include presentation graphics, color overlay capability, improved symbol scaling, and improved alternate test. Upgrade: $50. Support for the HP7470A plotter released as separate module. $50.

Δ Developed to assist realtors, brokers, bankers, and accountants, *Competitive Edge* can quickly analyze complex purchase offers and determine the best terms from the seller's point of view. From **Softronics** (100 South King Street, Seattle, WA 98104; 206-587-0688). The software allows a seller to compare offers, determine which offer generates the maximum after-tax cash flow, and so on. $500. Demo kit: $10.

Δ Simple diagnostic software that checks the performance of disk drives, the *Datalife Disk Drive Analyzer,* from **Data Encore** (585 North Mary Avenue, Sunnyvale, CA 94086; 408-720-7400), runs four tests in less than three minutes to check head alignment, disk clamping, write/read accuracy, and disk speed. $39.95.

Δ People with no programming skills or previous authoring experience

Introduce yourself to the IBM/PC with "HELLO CHARLIE"



Would you like to see what your new IBM/PC can do without spending hundreds of dollars on software? "HELLO CHARLIE" gives you a smorgasbord of some of the most popular types of programs available for the PC. Entertainment, graphics, spreadsheet, typing tutor, it's all there. Some of the programs include:

■ **A Graphics Tool**— Turn your PC into a drawing board with "HELLO CHARLIE'S" graphics program. Create, scale, transfer and rotate pictures on your display. Then print hard copies with screen print routines for most of the popular dot matrix printers.

■ **A Typing Tutor**—Highly interactive, self paced tutor that teaches you how to type correctly and increase your typing speed.

■ **A Financial Tool**—"HELLO CHARLIE" includes Calc-86™, a powerful spreadsheet program that's functionally comparable to VisiCalc® and SuperCalc®. With its many powerful functions and commands, Calc-86™ is the only spreadsheet you'll ever need for your personal and business planning needs.

■ **An Entertainment Package**—Six exciting games, ranging from mindbenders like Othello and Life to action games like Millibug and Chomper.

"HELLO CHARLIE" turns your PC into the focal point for education and amusement. So say hello to your IBM/PC—go to your local IBM dealer and say "HELLO CHARLIE" today.

$99.95

ORION SOFTWARE

Orion Software, Inc.
P.O. Box 2488
Auburn, AL 36831
To order direct call:
800-821-8088

HELLO CHARLIE

can create, present, and administer computer-aided instruction with the *Personal Computer Instructional System* from **IBM** (Box 1328, Boca Raton, FL 33432; 800-447-4700). With the software, courses can be authored on an IBM mainframe, then offered on the PC and vice versa. Authoring system: $525. Presentation system: $85. Administrative system: $400.

Δ **Alpha Computer Service** (Box 2517, Cypress, CA 90630; 714-894-6808) has announced *ForLib-Plus*, a set of three assembly-coded libraries, subroutines, and demo programs to support the Fortran library found on your Fortran distribution disk. The three libraries contain support for graphics, communications, file handling, and disk support. Requires 192K (256K recommended), Microsoft Fortran Compiler version 3.1 or the IBM Fortran Compiler, and DOS 1.1 and 2.0. $69.95.

Δ Product news from **Chalk Board** (3772 Pleasantdale Road, Atlanta, GA 30340; 404-496-0101): *The Power Pad* is a twelve-inch-square, touch-sensitive input tablet that can replace the keyboard with the help of a Mylar overlay. Software for a variety of applications is included along with overlays of an artist's canvas, a piano keyboard, a game board, and Logo. $99.95. Δ *Leonardo's Library* is a series of entertaining educational software packages for learning music, mathematics, visual arts, science, language arts, and social studies at various levels. $24.95 to $49.95.

Δ Identical to its mainframe namesake, *PC/Focus*, from **Tymshare** (20705 Valley Green Drive, Cupertino, CA 95014; 408-446-6000), is a database management system for application development that replaces conventional programming with English-language sentences. Can be used to create applications for mainframes as well as the PC.

Δ A color/graphics software package that allows you to produce graphics and animation in twenty-six colors is available from **Hypergraphics** (1908 Stonegate Drive, Denton, TX 76205; 214-783-9900). The *Hypergraphics* system can be used as an authoring language for tutorials, stand-up sales presentations, creating marketing disks, and more. It can also pick up a buffer-stored screen from another software package and convert it to color graphics. Requires 128K. $395.

Δ On-line news and investment information from the Associated Press can be accessed through the IBM Information Network without any fees or line-usage charges with the introduction of the *ByteLine* service from **Compubyte** (313 Airport Road, Naples, FL 33942; 813-793-1656). World, national, and regional news, charts, special reports, and software programs are also transmitted through the service. $295.

Δ *The Professional Accountant Software System* is available from **Plenary Systems** (9669 Wendell Road, Dallas, TX 75243; 214-343-9901). The three software packages that make up the system include a client write-up and financial reporting module, time and charges module, and a comprehensive 1040 tax preparation and planning module. Requires RM/Cobol run time. Module I and II: $1,995. Federal tax module: $895. State: $495.

Δ Two printers can be run from the same computer with *Le Switch*, from **Renaissance Technology** (1070 Shary Circle, Concord, CA 94518; 415-676-5757). A modem can be installed in one of the unit's serial ports. Parallel: $155. Serial: $125.

Δ *Winning on Wall Street* is stock market decision-support software designed for investors and traders. The three-module system from **Summa Software** (Box 2046, Beaverton, OR 97075; 503-644-3212) includes *Trader's Data Manager*, which provides reports and graphs, $200; *Trader's Forecaster*, a forecasting and technical analysis tool kit, $250; and *Trader's Accountant*, which handles all types of stocks and securities, $350. Complete system: $700.

Δ **Opportunities for Learning** (8950 Lurline Avenue, Chatsworth, CA 91311; 213-341-2535) announces the new edition of *Selected Microcomputer Software*, a catalog of educational courseware for instruction, administration, and testing. Covers a range of grade and skill levels from primary through college. Includes programs for several makes of computers. Free. ▲

ThinkTank™

IS NEW AND UNIQUE. IT IS VERY USEFUL. YOU SHOULD KNOW ABOUT IT. READ THIS CAREFULLY.

WHAT THE PROGRAM IS

ThinkTank is an 'idea processor' for your IBM PC personal computer, an outline editor with powerful features. It allows you to deal with the thoughts in your mind by providing a simple, versatile way of outlining, arranging and modifying text. Integrated into ThinkTank is a powerful, easy to use word processor.

HOW ThinkTank WORKS

You enter topics in 'headline' form—and then indent headlines under each other to create an outline format. With single keystrokes you change your outline to reflect new thoughts. You can put a page of text under your headline—when you move the headline the text goes with it.

You can sort alphabetically, or search for key words or phrases. You can 'expand' to reveal hidden detail or 'collapse' any part on the screen to see the big picture. You can edit and print it out.

WHO CAN USE IT

Virtually anyone with a personal computer needs ThinkTank. It is being used for project management, organizing, writing anything, notetaking, bibliographies, lesson planning, scripting, action lists, and brainstorming. It can be your file cabinet, bulletin board, administrative assistant. Anyone who uses his mind to get through life will find ThinkTank an extremely valuable tool.

USERS AND REVIEWERS LOVE IT

The New York Times: "ThinkTank is so easy to use, and so relatively error proof that even a first-timer feels as if he's in charge of the computer, instead of the other way around. And being in charge of the computer is what enables you to do with it things you may never have thought of doing before."

InfoWorld: "Your screen becomes a dynamic arena for your ideas," and "an amazing tool that emphasizes the power of the microcomputer."

One customer wrote: "I use ThinkTank for more purposes than I have any other program in the five years I've owned a computer."

TRY OUT ThinkTank. YOU WILL LOVE IT TOO.

Suggested Retail Price $195 (IBM PC)

ThinkTank is available for IBM PC, Compaq, and 100% compatibles. You'll need two DSDD floppies or a fixed disk and one floppy; 256K RAM. Also available for Apple computers.

ThinkTank and "the first idea processor" are trademarks of Living Videotext, Inc. Apple is a registered trademark of Apple Computer, Inc. IBM and IBM PC are registered trademarks of International Business Machines Corporation.

Living Videotext Inc., 1000 Elwell Court, Palo Alto, CA 94303 (415) 964-6300

THINKTANK
The First Idea Processor

BOARDS AND BUSES

by Kevin Goldstein

Levels of XT Compatibility

The ultimate test of XT compatibility for an add-on hard disk is this: Does there exist a program, operating system, utility, operating environment, expansion card, peripheral, printer, disk, expansion chassis, and so on ad infinitum, that works—properly, with no errors or anomalies—with the XT but doesn't work with the add-on disk? If no such animal exists, then you've got yourself the functionality of an XT. If such a critter does exist, then the proposed configuration is to a greater or lesser degree not truly XT-compatible, and your ability to use standard software and hardware will be impaired accordingly.

One can identify four levels of hard-disk compatibility. Machines compatible at all four levels should pass our ultimate test of compatibility; others will show varying degrees of compatibility, depending on the level at which they fail.

Roughly speaking, the four levels of compatibility can be called: the board, BIOS, device driver, and out in the cold. Out in the cold is where you are when the manufacturer supplies a disk drive but offers no software or hardware with which to integrate the drive into the PC's software and hardware. Unless you're a determined software hacker, an engineering whiz, and a masochist (in which case you might as well start your own company), you needn't concern yourself with hardware supplied at this level, because you aren't going to buy it.

Moving up a level in compatibility we hit what, for want of a better name, we'll call the device-driver level. A manufacturer marketing a disk system compatible at this level supplies you with three major elements: the disk itself, complete with a disk-controller card (which may not be compatible with the IBM hard-disk controller card); a host-adapter card, which is the electrical and mechanical link between the disk and the PC (see the accompanying diagram and refer to last month's column for a description of the disk-to-computer interface); and a device driver, the program that provides the software link between the operating system, the operating system's clients, and the hard disk. (The "clients" of an operating system are application programs, such as word processors or spreadsheets.)

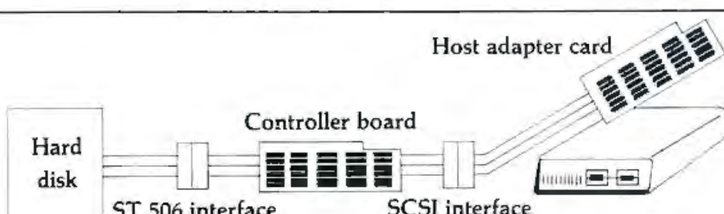
On the XT, the software necessary to manage the built-in hard disk resides in ROM. The ordinary (non-XT) PC lacks that little piece of software, so an external disk cannot call on routines within the PC's ROM. That's where system integrators come in: They supply the software routines necessary to construct an interface between the PC and a hard disk.

When an application program needs to write data to a mass storage device (which for our purposes means a hard disk or floppy, but could just as well be RAM or bubble memory configured to look like a disk), the program tells the operating system where the data to be written are and then lets the operating system make the transfer.

DOS 1.1 and earlier incarnations came with a standard set of routines to write data to floppy disks or to the cassette port. DOS 1.1 knew nothing of hard disks, however, so if you wished to use a hard disk or other non-IBM peripheral on your machine, you had to patch some code into DOS that told it how to access the new peripheral.

Patching DOS is a pretty sloppy way around the problem; every manufacturer will slip patches into slightly different places in the system. A patch literally creates a new, nonstandard DOS, and, as you can imagine, sooner or later a nonstandard DOS is going to be incompatible with something—an application program, perhaps, or yet another new peripheral.

DOS 2.0, in contrast, provides for the orderly addition of routines to itself. The new routines are called *installable device drivers* and are one of the principal reasons that DOS 2.0 is so much more powerful than its predecessor. Don't be misled by the fancy name; a device driver is simply a short program or routine by which DOS can control the interface



Typical External Hard Disk—PC Interface

Would you like to get in the picture?

. . . then join the FLEET!

FEATURES OF STAR FLEET I
The War Begins

- Full color battle animation
- Enemy tactical and strategic movement
- Thirteen starship systems simulated
- More than 20 commands available
- Sophisticated weapons auto-fire
- Damage control for repairing systems
- Tractor beam, transporters and space marines for capturing and towing enemy vessels
- Internal security to protect vital systems and deal with enemy intruders
- Selection of 36 starships for command
- Four independent defensive shields
- Sound can be toggled on/off
- Comprehensive **Officer's Manual**
- Quick Reference Card
- Much, much more ... !!!

STAR FLEET I - *The War Begins*, is the first in a series of advanced strategic programs designed with the **FLEET** concept. You will become an officer of **STAR FLEET**, and compete with friends to progress from a rookie cadet in the Academy to the illustrious rank of Admiral. You will be given more challenging missions as your abilities develop, and bestowed with decorations and awards for outstanding performances along the way. Promotions and awards are stored in your service record on your playing diskette for view by all.

STAR FLEET games combine color, animation, and music with high level strategy in an exciting, mind-stimulating package. The spiraling levels of play, exacting attention to detail and thorough documentation (the **STAR FLEET I Officer's Manual** contains over 100 pages) ensures **FLEET** members years of glorious battles.

STAR FLEET I is currently available for the IBM* PC or equivalent with at least 96K RAM, one disk drive, DOS, BASICA and an 80-column display with either the monochrome or color/graphics adapter.

CYGNUS
the bright new star in software

presents

STAR FLEET I™

THE WAR BEGINS!™

You have completed your training at the Academy. Now is the time to see if those long hours in the simulator pay off as you accept command of one of the most powerful fighting vessels in the known universe--the Invincible Class heavy cruiser. Your mission: defend the outer regions of the United Galactic Alliance against attack by hordes of ruthless Krellans and Zaldrons. You engage main engines and enter hyperspace confident of success. However, you arrive in the Deneb IV region to suddenly find yourself greatly outnumbered ...

CAPTAIN'S LOG, U.G.A.S. *ATLANTIS*

Date 5106.2: *We emerged from hyperspace surrounded by four Krellan destroyers. GENERAL QUARTERS! We immediately engaged battle and fired off five torpedoes.*

Date 5106.3: *We destroyed two Krellan vessels and disabled a third, but a Zaldron warship arrived. We are laying mines since their invisibility screen makes our sensors useless.*

Date 5106.6: *Sensors indicate a large explosion. The Zaldron has struck a mine! Our tractor beam has the disabled Krellan ship in tow and our space marines are being transported aboard to effect capture.*

Date 5107.0: *Our boarding party was repulsed with heavy casualties. INTRUDER ALERT! The Krellans retaliated by beaming a sabateur aboard. An explosion on Deck 7 damaged our primary life support system. Have initiated a search and put Deck 7 on max security.*

Date 5107.6: *Krellan vessel finally captured by the marines and prisoners were taken. Received top priority orders to rescue Starbase 3 which is under attack by a Krellan fleet. Crew at battle stations--engaging hyperdrive.*

Date 5107.9: *COLLISION! Course intercepted by Krellan. POINT BLANK HIT! Have lost shield 1 and suffered massive damage and casualties. Engineering reports our backup life support system is failing. Cannot survive another attack or reach starbase! We must escape ... but HOW?!*

Membership in the **FLEET** is available for only $49.95, suggested retail price, at your computer software store, or send check or money order, plus $2.00 for postage and handling, to: **CYGNUS, STAR FLEET I**, P.O. Box 57825, Webster, Texas. 77598 Tel: (713)486-4163 Dealer inquiries welcome.

*Registered trademark of International Business Machines Corporation

between itself and a nonstandard peripheral (a device).

Once you understand what a device driver is, you can start to appreciate how the shell—or the user interface to the operating system, which in the case of PC-DOS resides in a file called Command.com—is the tip of the iceberg. The real power of the operating system lurks beneath.

You might wonder how DOS 2.0 manages to implement a standard method of adding various-sized pieces of code to itself. The answer is pretty simple and relies on a concept known as a *linked list*. To understand this concept, assume you're writing out a grocery list, numbering each item as you go along. When you're done, you realize that if you rearrange the list to match the order of the groceries on the shelves, you'll save some steps at the store.

You can retype the list in the proper order (assuming you know how the store stocks its shelves), but here's a cleverer approach: Instead of retyping, just write a number after each item on your list. The number you write is the line number of the item that would come next in order if you had retyped the list. In other words, each item in your list is followed by a pointer to the item that follows it—in the order of your grocer's stock. By following this second approach, you create a linked list.

Notice some advantages of the linked list. If the store rearranges its stock, all you need do is change a few "next-item" numbers—in other words, you can simply redirect the pointers. You can add items to the bottom of the list; if the items are found early in your walk through the store, changing just a couple of pointers will quickly insert them into the right "logical" place in your list.

This, incidentally, illustrates how the computer world uses the term *logical*, in contrast to the term *physical*. The item you add to the bottom of the list is physically the last item; by rearranging the links or pointers in the manner just described, you put the newcomer logically at the appropriate place in the list.

Compare this approach with the *assign* statement in DOS 2.0, which does really nothing more than provide a couple of pointers. *Assign b=a* means physical drive A will be called drive B. Or, to put it another way, physical drive A becomes logical drive B.

A linked list is what DOS uses to arrange the device drivers you and DOS install at start-up time. When DOS is asked to write a block of data to drive D, for example, it simply bounces through the links of its device driver list until it finds the device called D, and then it calls the piece of code stored under the name D (the code for the driver itself) to control the actual data transfer.

DOS 2.0 comes with drivers for such devices as the console (Con:), floppy disks, the printer (Prn:), and so on. If you were to write another device driver with the name Con:, and install it by putting the statement

Device=con.com

in the configuration file, your new device driver would be linked to the list prior to the preexisting driver for Con. Since DOS stops searching the list when it finds the first occurrence of the name it's looking for, you could write your own device driver for the console and, simply by naming it Con:, replace the existing driver.

In any case, that's how DOS 2.0 can add a variable number of variable-sized pieces of code, without creating a nonstandard DOS and associated compatibility problems.

Let's see if we can unstack all the interrupts and get back to the original topic, which was levels of compatibility for hard disks. We were talking about the second level, that level of compatibility that results from interfacing via an installable device driver.

Compatibility at the device-driver level is so-so. Application programs that don't bypass DOS but direct all their requests for disk storage to the operating system will have no problem running with a disk that's compatible at this level. (It is to be fervently hoped that all software publishers will write legal programs

that don't jump around DOS. Programs that try to outsmart DOS will be out in the cold when operating environments such as VisiOn or Windows get popular or when the next version of DOS, which allegedly will include multitasking, makes its debut.)

A device driver is really just an extension of DOS. Loading an operating system other than DOS wipes out DOS, and with it the device driver and any access you once had to your hard disk. Therefore, the only way you can use foreign operating systems with such a disk is to depend on the disk's system integrator to support the foreign system. That's something you don't want to depend on if you simply must have some esoteric operating system.

As mentioned earlier, the ROM that comes with the XT includes a device driver for the hard disk. The driver is one of many routines that make up the BIOS, or basic input output services. A few disks, such as the Maynard internally installed hard disk mentioned last month (and possibly to be reviewed in the next column . . . or the next two), come complete with an EPROM that contains a driver for the disk. In such devices, the presence of the driver no longer depends on its being loaded via DOS, so such devices frequently will work with foreign operating systems.

If "frequently will work" sounds a little vague, that's because it's meant to be. Whether a system really will work depends on two factors: whether the BIOS is written properly, with no bugs (a fairly gargantuan task) and whether the foreign operating system accesses mass storage via BIOS routines or writes directly to the control registers on the disk-controller board. As an example, a program like Peter Norton's hard-disk utility, written specifically for the XT, should run okay on a system compatible at the device-driver level, since his utility accesses the disk via the BIOS. Venturcom's operating system Venix, on the other hand, writes directly to the disk-controller board, so it will not work with disks compatible at this level.

That brings us to the highest level of compatibility: functionally equivalent controller boards. The Winchester controller board in the XT is a nifty little item manufactured by Xebec. Like any other piece of "intelligent" hardware, it must be talked to in the limited language it understands. For example, to read data, the particular bit pattern specifying a read operation must be sent to the I/O port wired to the Xebec card. A foreign operating system that end-runs the BIOS and writes directly to the hardware and that thinks it's working with an XT, is going to send the bit pattern the Xebec wants for a read operation, regardless of whether the hard disk is controlled by a Xebec card or one knitted by your mother-in-law.

Any OEM or system integrator can go out and buy Xebec controller cards to use with its hard disk. That's what system integration is all about. Or system integrators can design their own card, making it a functional clone of the Xebec. Or a system integrator could buy a card such as the ACB-2002 from Adaptec (another major manufacturer of disk-controller cards), which is claimed to be 100 percent XT-compatible.

The ACB-2000 series from Adaptec is interesting, because it appears to be the first fully XT-compatible OEM controller board from a major competitor of Xebec. When introduced in late November, the board came with an empty socket for an EPROM—which means that system integrators would have to write their own BIOS. IBM's XT BIOS is copyrighted, so manufacturers can't just copy IBM's version and sell it with their own system and, as mentioned earlier, designing a BIOS with no bugs can be a real bear. For that reason, Adaptec plans to supply a prewritten (presumably XT-compatible) BIOS with its board—which might be available by the time you read this. The advent of such a board is noteworthy, since it should eventually drive down the price and increase the availability of truly XT-compatible add-on hard disks. ▲

The personal, portable daisywheel printer.

Only $599.

For the first time, your letter-quality printer can be used almost anywhere! Bring the new Transtar 120 with you to work, to school, and home again! Conveniently weighing in at less than 19 pounds, it generates unrivaled print quality and is the size of a standard briefcase. The new 120 is so light, so small, that you <u>can</u> take it with you!

Remarkably, the new $599 Transtar 120 is "plug and go" compatible with the best-selling word processing programs. Just plug the 120 into your personal computer and watch this precision printer purr along at 14 cps Shannon text speed producing superscript, subscript, underlining and a true boldface. Even using letterhead is now a breeze with the 120's automatic single sheet loading!

Don't worry about durability: it's a tough little machine. It joins the highly reliable family of Transtar printers with a failure rate that's the envy of the industry: less than 1%. Should your 120 ever need repair, a nationwide network of authorized service centers stands ready for speedy repair on your six-month end-user warranty.

Just think of it: everything you want in a letter-quality printer...anywhere you want it. Only $599.

Transtar

P.O. Box C-96975, Bellevue, Washington 98009

COMDEX
THE RITE OF AUTUMN

by John Socha

For the cab drivers, bellhops, doormen, and travel agents of Las Vegas, Nevada, the last week of November is the busiest of the year. It's the week that nearly a hundred thousand people flock to Glittertown for the computer industry's fall rite: Comdex.

What's Comdex and who are these pilgrims?

Comdex is the preeminent trade show for this industry. The "D" in the show's name stands for *dealers*, which means retailers and distributors. These two groups come to Comdex to see what's new, and plenty of software and hardware companies are willing to show them. Vendors try to find distributors for their wares, while distributors prowl the floors in search of that one product they can "discover" and turn into a gold-mine hit. OEMs do their thing, the press does its, and the whole business is hectic and crazy. It's an exciting week.

Some observers declared this year's show not so exciting. They claimed, correctly, that most of the new products were "me too" items—another calc-alike, another database program, or (yawn) another word processor. Even so, the show did bring forth some new products of interest. So here's a collage of Comdex impressions that should give you an idea of things to come.

PC-Compatibles. At last spring's Comdex in Atlanta, Quadram introduced Quadlink, its answer to computer owners who want both an IBM PC and an Apple II. Quadlink was a fine feat of engineering, but it didn't do much for PC owners who use machines other than the Apple—the Osborne or Kaypro, for example. Isn't there some way to build a general-purpose computer. Zads, no, you say?

Zads, yes. Take another look at a new entry, dubbed the Dimension, from Micro Craft Corporation. Their ads—which have been appearing in magazines like *Byte* and drawing thousands of inquiries—claim they sell "A single personal computer able to handle Apple, IBM, TRS-80, Unix, and CP/M-based software." Sounds too good to be true. In fact, it sounds impossible.

Nonetheless, the company seems to have

Introducing a graphics system to meet your every need, whether it's creating sophisticated presentations that integrate screens from your favorite programs — or expressing yourself with freeform drawings.

What Lotus™ 1-2-3 has done for spreadsheets, Executive Picture Show™ is doing for graphics.

Ordinary business graphics programs are fine as far as they go. But they are hardly the stuff of interesting business presentations.

If yon need the ability to integrate and modify screens from other programs, such as Lotns 1-2-3, dBASE II®, CREATABASE, and WordStar™ — or just give free rein to your artistic side with free-form drawings — you've probably been frustated on both connts.

Now there's a business graphics system that gives you these capabilities, plus the usual line, bar, 3D bar, horizontal bar, and surface and pie charts.
It's the Execntive Picture Show and it's long on capabilities where other programs fall short.

With Executive Picture Show you can create:

- free-form graphics
- business graphics
- slide show presentatiuns
- animated presentations

Presentations to keep viewers on the edge of their seats.

Not only does Execntive Picture Show accept screens from other programs, it lets you integrate them into your presentation where and when you want them. Then you can add the sound, motion, and color that insures a captive andience. Dropping in your company logo or making bar graphs take form right before your viewer's eyes is easy as pressing a few keys.

Executive Picture Show allows you to use both yonr IBM® monochrome and color monitors during your presentation. This means you can show a spreadsheet on your monochrome monitor, while a graph or drawing is formed on your color display.

Interactive presentations.

The Executive Picture Show was designed with you and the audience in mind. Not only does it allow you to design a moving presentation, it gives your viewers a chance to respond with more than enthusiastic reviews. They can actually input their responses so the program — and you — can act on their input.

Easy to use.

We had the business executive in mind when we designed this tool. That's why Executive Picture Show cuts throngh all the "computer-ese" to simplify instruction and prompts you throughout the program. We've reduced many steps to single key commands and included a handy reference card to help you get your show on the road.

Executive Picture Show is playing at a computer store near you for just $195. If you want to preview this program, contact PCsoftware of San Diego directly for a demonstration disk and documentation priced at $30.

Requires: Graphics adapter and display
128K RAM
2 disk drives or hard disk
IBM PC or IBM XT

Lotus 1-2-3, CREATABASE, and WordStar are trademarks of Lotus, PCsoftware of San Diego, and Micropro International Corp., respectively. dBASE II and IBM are registered trademarks of Ashton-Tate and International Business Machines, Inc.

Dealer inquiries invited.
VISA and Mastercard accepted.

Dealer orders contact:

Micro D	(CA)	800-432-3129
	(Nat.)	800-854-6801
Software Distributors	(CA)	800-252-4025
	(Nat.)	800-421-0814
Vitek	(CA)	800-237-7290
	(Nat.)	800-237-3443

Also available through Computerland Corporate.

PCsoftware™
PCsoftware of San Diego
Suite 416
9120 Gramercy Drive
San Diego, CA 92123
(619) 571-0981

delivered. The Dimension is probably more compatible with the PC than a lot of other machines that make that claim, yet it's also compatible with the Apple II, TRS-80, Kaypro, and Osborne. Micro Craft showed the Dimension running *Microsoft Flight Simulator*, a program that doesn't run on some other PC-compatibles.

And it's not done with mirrors. The heart of the Dimension is a 68000 microprocessor that handles much of the workload; extra boards provide for compatibility with other computers. By adding an 8086 board plus some software for the 68000, you get the PC as well as the 68000. Another board gives you a Z-80 microprocessor for emulating many of the popular CP/M-80 computers. By changing the software running on the 68000, you can switch between an Osborne and a Kaypro, just like that. Add a 6512 board and some more software, and you've got an Apple II. And it really does behave like an Apple II; it even runs some copy-protected games.

How much is this dream machine? Just $3,995 with two 5-1/4-inch disk drives and 128K, plus CP/M-68K, Basic, C, and an assembler for the 68000. The processor boards sell for $495 each. Expect to hear more about the Dimension in the future.

Other computer manufacturers across the show floor were boasting that they had the most alike-working PC-work-alike around. New compatibles abounded, many offering little more than the IBM. Companies such as Sperry, ITT, and Panasonic all introduced PC-compatible computers. But one company introduced a multiuser PC.

(A what?)

North Star announced the North Star Dimension (oops, not related to the Dimension computer). The North Star Dimension has a central module, with a fifteen-megabyte hard disk, that can support up to twelve users. You can add new users just by attaching a new board to the central module, so each user has his own memory and 8088 microprocessor in the central module. All users share the central resources, such as the hard disk and printer. This system offers considerable savings over more traditional local-area networking arrangements. Unfortunately, each workstation must be within 300 feet of the central module, and the cables are much thicker than Ethernet cables.

Another company, PC Technologies, has taken a different approach. Rather than placing all the microprocessor boards in the central module, PC Technologies has chosen to build PC-compatible computers sans disk drives and link them together with Ethernet. This approach allows you to mix IBM PCs with PC Technologies workstations, and the cables can be longer than 300 feet.

On a totally different front, Otrona introduced a new version of its Attache computer. The old Attache was an unusually small portable CP/M-80 computer. The new version has sprouted (what else?) an 8088 for PC compatibility. The torrent of PC clones is becoming tiresome, but the Attache is probably the smallest and lightest of the lot, weighing in at eighteen pounds. Warning: at $3,795 it isn't cheap.

Printers. Last year's rage in dot-matrix printers was the so-called correspondence-quality printer. Correspondence quality means you still see dots but the characters are a heckuva lot nicer than the those produced on the old Epsons. That's dandy all right, but correspondence quality is still not letter quality. Things are different now.

At Comdex this year, a number of companies introduced printers capable of producing text that looks almost like it was typed on a typewriter (almost because you *can* see the dots if you look carefully). By far the most impressive was the P2 Pinwriter (just one letter short of Spinwriter), from NEC.

The P2 features an eighteen-pin print head with overlapping wires. In high-speed, or draft, mode, the P2 churns out 180 readable characters per second; these characters are obviously dot-formed and look much like Epson's. If you don't mind slowing down a bit, you can get so-called correspondence-quality characters at a mere ninety per second. But now switch to the dual-pass mode, which NEC calls "near letter quality." The output is outstanding, and, at thirty characters per second, the P2 in this mode is almost as fast as its big brother, the 3550 Spinwriter.

But the P2 is a lot more versatile. It can print graphics at 180 dots per inch. Furthermore, those beautiful characters are downloadable, which means that you can define your own set. At $799 for a P2 Pinwriter, it's hard to go wrong, especially when you consider that traditional letter-quality printers are more expensive, sometimes slower, and sometimes heavier. For another $15 you can buy a sheet guide. Just place a single sheet of paper in the guide, pull a lever, and the P2 automatically loads the sheet without your having to align it in the printer. Expect to find the P2 in stores in February.

Several other companies—among them Epson, Fujitsu, and Toshiba—were showing twenty-four-wire printers with near-letter-quality printing.

The Toshiba P1350, which has been around for a number of months, is the least impressive. What it lacks in quality, however, it makes up for in speed. The Toshiba pounds out high-quality characters at an astounding one hundred per second. Unfortunately, like the Epson and Fujitsu, the P1350 sells for about $2,000—more than twice the price of the NEC.

Fujitsu was so proud of its DPL24 that the

Instant one-button color printing.

Press here.

It's just that easy! Any time you want to print what's on your Apple's screen just hit the copy button on your Transtar 315 color printer with our PICS card installed, and it's done! No special programming, no lengthy code sequences, no need to exit your program! Just press the button and it prints!

By adding the optional PICS card to your $599 Transtar 315 color printer, you've opened up a whole new world of easy color printing. For the first time ever, our PICS parallel interface card enables you to screendump virtually any program -- graphics, charts, games -- even copy-protected software! Specially designed only for the Apple II, II+, IIe, and Franklin computers, the Transtar 315 PICS card does the work of a parallel card and a lot more and costs only $119.95.

At the push of a button, Transtar's innovative new 4-color diagonal ribbon will print up to 7 colors and more than 30 shades in a single pass.

The 315 is precision-built to exacting standards by Seikosha, the most experienced company of the famous Seiko group-- recognized worldwide for quality and dependability. In fact, one of the nicest things about Transtar's 6-month warranty on parts and labor is that you'll probably never use it!

Innovative, inexpensive, dependable, easy: the Transtar 315. Color printing has never looked so good!

Only $599.

*PICS cards are currently available for Apples and Franklins. PICS cards for other computers will be available in the future.

Transtar
A Vivitar. Computer Product

P.O. Box C-96975, Bellevue, WA 98009

company was handing out magnifying glasses to help you find the dots. This printer does in fact pass the invisible-dot test nicely, but, alas, it will sell for about $2,000 (when it becomes available in April or May). Other details about the DPL24 are scarce, although Fujitsu did say its printer will produce letter-quality output at eighty characters per second.

With an impressive number of new features—such as italics and half-height subscripts and superscripts—Epson's new LQ-1500 goes beyond the realm of letter quality. Unfortunately, its characters are more noticeably dotted than those of the other printers. At about $1,500, however, the LQ-1500 is likely to become popular (when it becomes available).

Lisa and Sisters. Windows are here. VisiCorp has been talking about VisiOn, its answer to Apple's Lisa computer, for over a year now. But so far it's been all talk, no delivery. Well, VisiOn has arrived at last—for $495.

That's the price for VisiOn alone. You still have to put something in the window, and in VisiOn's case that means specially written application programs. With *VisiOn Calc*, *VisiOn Word*, and *VisiOn Whatever* priced at about $375 each, the package is costly. And don't forget the hard disk, 512K, and mouse. There's been some doubt expressed about how well Visi will catch On, but VisiCorp's CEO Dan Fylstra was saying at the show that the company has a back-log of thirty thousand orders.

The doubt concerning VisiCorp's prospects stems from the fact that the application programs that we're all running now will not run in the VisiOn environment. Such is not the case with certain other window systems.

Quarterdeck's Desq, introduced last spring and available now for about $400, works with almost any software package now available for the PC under PC-DOS. True, Desq doesn't offer the power of VisiOn, but it's more attractive to users who don't want to buy a whole lot of new software.

Welcome Microsoft to the scene with Microsoft Windows. These are portholes with a difference. Microsoft doesn't overlap its windows; instead, it uses a scheme it calls tiling, wherein the screen areas (windows) adjust in size so that together they fill the screen.

For several reasons, Microsoft looks like a winner in the window war.

To begin with, there's price. Microsoft won't be selling Windows directly to end users, any more than they sell MS-DOS directly to end users. Instead they'll sell their environment to computer companies such as Radio Shack and Compaq. Will IBM sell the system? They're not saying. But if they do, expect them to sell it for about $150.

Windows, moreover, has much of VisiOn's power and Desq's versatility. Programs that don't know about Windows will run without change but won't use the full power of the environment. Programs written for Windows, on the other hand, will be able to take advantage of its graphics, pop-up menus, and mouse support, as well as its channels for data interchange with other programs.

So what do you need to run Windows? A stock PC, one floppy disk drive, a mouse, and a graphics adapter—nothing more. And since Microsoft wrote Windows, one can assume that it works with their nonexistent, unacknowledged DOS 3.0 and their existing Xenix operating systems. Don't hold your breath, though; Windows won't be available until the second quarter of this year.

Mass Storage. Ever imagine squeezing three megabytes onto a floppy disk? Soon you'll be able to. Amlyn has introduced a 3.2 megabyte, 5-1/4-inch floppy disk drive. It uses floppy disks that have had a special alignment track added for extra reliability. According to Amlyn, these megafloppies are just as reliable as the standard 360K variety.

And you'll be able to back up a ten-megabyte hard disk onto about three super-floppies. Amlyn says a complete 3.2 megabyte floppy with controller should sell for less than $1,000.

Cartridge hard disks are even nicer, offering hard-disk speed and a full five megabytes on a single cartridge. Syquest says it's finally delivering reliable cartridge systems (to OEMs). Last year the company was having all sorts of trouble; it seemed that if you wrote a cartridge on one drive, you couldn't read it on another. Now the company claims those problems are solved, so expect cartridge drives to be available soon. How much? They'll cost about as much as a five-megabyte Winchester.

On the blue-sky scene, Panasonic has introduced a number of videodisc recorders capable of storing ten thousand still photographs on a single eight-inch videodisc. Soon similar systems may appear that will store computer data, putting perhaps as much as one *billion* bytes on a single disc. You won't be able to erase one of these videodiscs, but with that much storage, there'll be plenty to burn. Who'll be the first to put a major database on videodisc?

Random Doings. When Digital Research tried to capture a piece of the sixteen-bit market with CP/M-86, it got socked by Microsoft. But DRI may stage a comeback with Concurrent CP/M-86 (CCP/M-86). Concurrent CP/M-86 allows you to run four different programs at any one time; that's nice in theory, but it doesn't do you much good if all your programs run under MS-DOS.

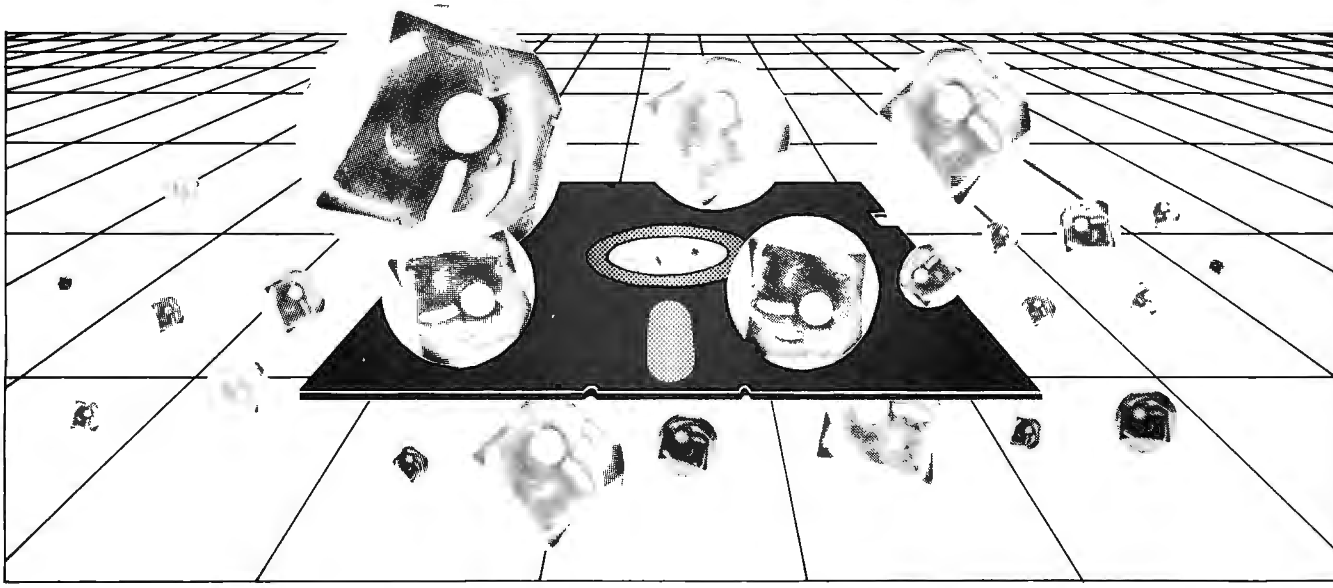
But imagine CCP/M-86 with multiple windows (a la Lisa) *and an MS-DOS emulator.* With that setup you could run your MS-DOS programs four at a time under CCP/M-86 with no help from MS-DOS.

Rumor says that Digital Research has such a product in the works and near release. With Microsoft Windows on the horizon, it's not clear how well an upgraded CCP/M-86 would fare, but it would certainly have some chances.

Finally, Princeton Graphics Systems was showing off its new graphics system for the PC. Sporting a 640-by-400 resolution capability, the board writes exceedingly clear characters in its text mode and some nice graphics, too. The system is still under development, so details aren't firm. But it's apparent that PGS plans to make its board compatible with IBM's color/graphics adapter in addition to supplying the new, higher-resolution modes. PGS's monitor sells for $800, while the board should go for between $600 and $700 when it becomes available in the second quarter.

Rumors abounded at Comdex. The most interesting had to do with a rather well-known young software company. In the history of software, no company that's introduced an outstanding first product has followed with an equally impressive second product, much less a more impressive one. But the grapevine says that Lotus Development may soon release something even more awesome than *1-2-3*. Let's hope so. ▲

We Blow Bubbles Around Your Floppy Disc.



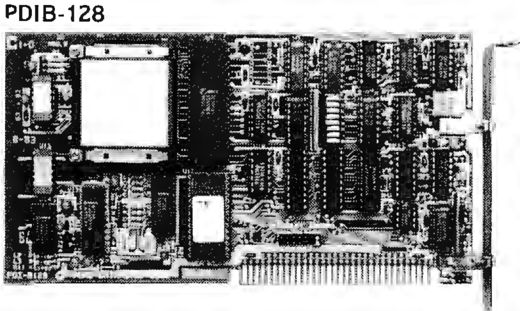
Charlie would have been blown away.

Bubble memory combines the best features of disk and solid state memory with extreme reliability. Unlike a floppy disk, it's entirely solid state with no moving parts. This makes it impervious to dust, dirt, fumes, and vibration. And, unlike normal memory, it doesn't lose data when the power goes off.

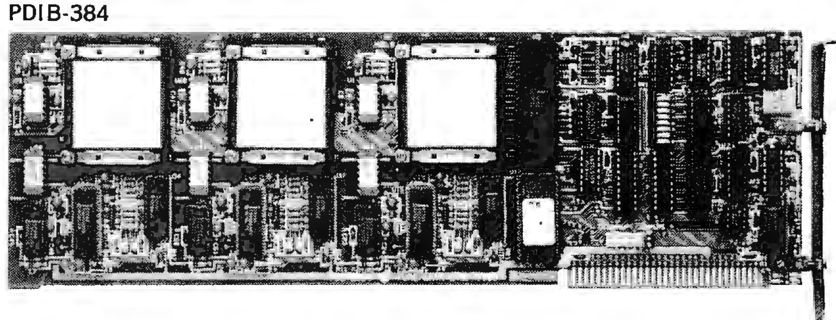
Now you can enjoy all the advantages of bubble memory combined with the famous Pure Data quality and reliability for your IBM PC or XT.

- PDIB-128 provides 128 KBytes in one slot
- PDIB-384 provides 384 KBytes in one slot
- Not affected by power failure
- Faster than a floppy disk
- Extremely reliable
- Standard DOS 2.0 disk-type device
- Compatible with all DOS software
- No patching of any system files
- Password option for computer and/or bubble
- Password cannot be bypassed by software
- Comprehensive diagnostics and other utilities
- Fully illustrated installation and operation manual
- Technical support hot-line
- Guaranteed 48 hour service
- Bubble uses 2 I/O addresses and no memory space
- Interrupts are supported but not required
- DMA is supported but not required
- On-board EPROM socket
- Pure Data quality and reliability
- No moving parts

PDIB-128



PDIB-384



Pure Data Ltd
950 Denison Street, #17,
Markham, Ontario, Canada L3R 3K5
☎ (416) 475-2424 ☎ (416) 498-1616

Pure Data products are available through.

OnTrak DATA SYSTEMS
1335 Valwood Parkway, Suite 108,
Carrollton, Texas 75006
Telephone (214) 620-8000
P.O. Box 815155, Dallas, Texas 75381

Highland Computer Corporation, Keysoft International, Computerland, Computer Innovations, Compugroup, ECOSEA Technologies and others.

PYXEL Presentations
Maximum Quality...Minimum Effort

PYXEL Visuals is a no-nonsense method for producing your presentations.

When you need:
- Large, bold text in a variety of sizes;
- Lines and illustrations;
- On-screen presentations as well as overhead transparencies;

PYXEL Visuals can do it.

When you need it fast and easy, PYXEL Visuals lets you

- Edit screens both before and after you save them on disk;
- See a directory of screens without leaving the program;
- Save a screen in a lot less than 16K of disk space (PYXEL Visuals uses only 1-3K);
- See what the hard copy will look like while you design it—not after you print it;
- Do all this without memorizing commands or constantly referring to your manual.

PYXEL Visuals does all this and more. Because you need something more...a low price. For only $98 you can be producing presentations (and signs and report covers) NOW. Without wasting time to figure out how.

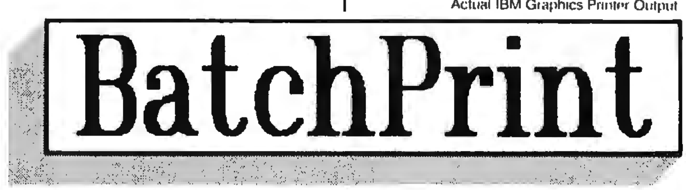
PYXEL Visuals gives you

- Horizontal or vertical hard copy
- Automatic or prompted on-screen presentations
- Character sizes 1/4″ to 1″ high
- Every character individually designed for best appearance
- HELP at the press of a key
- NO COMMANDS to learn —or forget
- Quick changes to screens and presentations

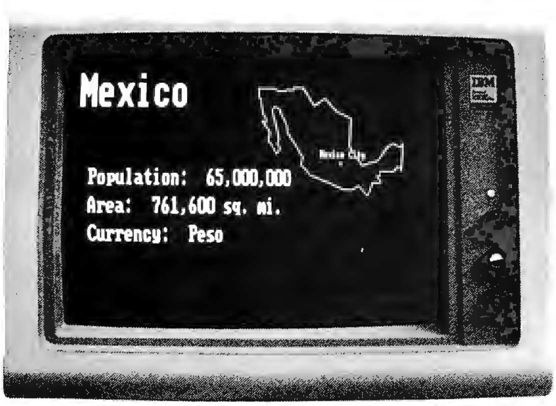
Our customers let us know they love it, but some want even more. So we created BATCHPRINT. For the person who needs even more quality, more control, and more freedom.

Our support is as simple as our software. We value our customers. You can call and get help with a problem. You'll receive updates and enhancements without rebuying the package. And, as a registered customer, you'll be offered new products at reduced prices.

Actual IBM Graphics Printer Output



BATCHPRINT prints PYXEL Visuals screens in one of four proportionally spaced print modes: high resolution bold, high resolution light, low resolution bold, and low resolution light.



You choose the spacing between screens on the hard copy so you can print short pages and very long pages. Of course, with a name like BATCHPRINT, you can print up to 42 screens while you get a second cup of coffee.

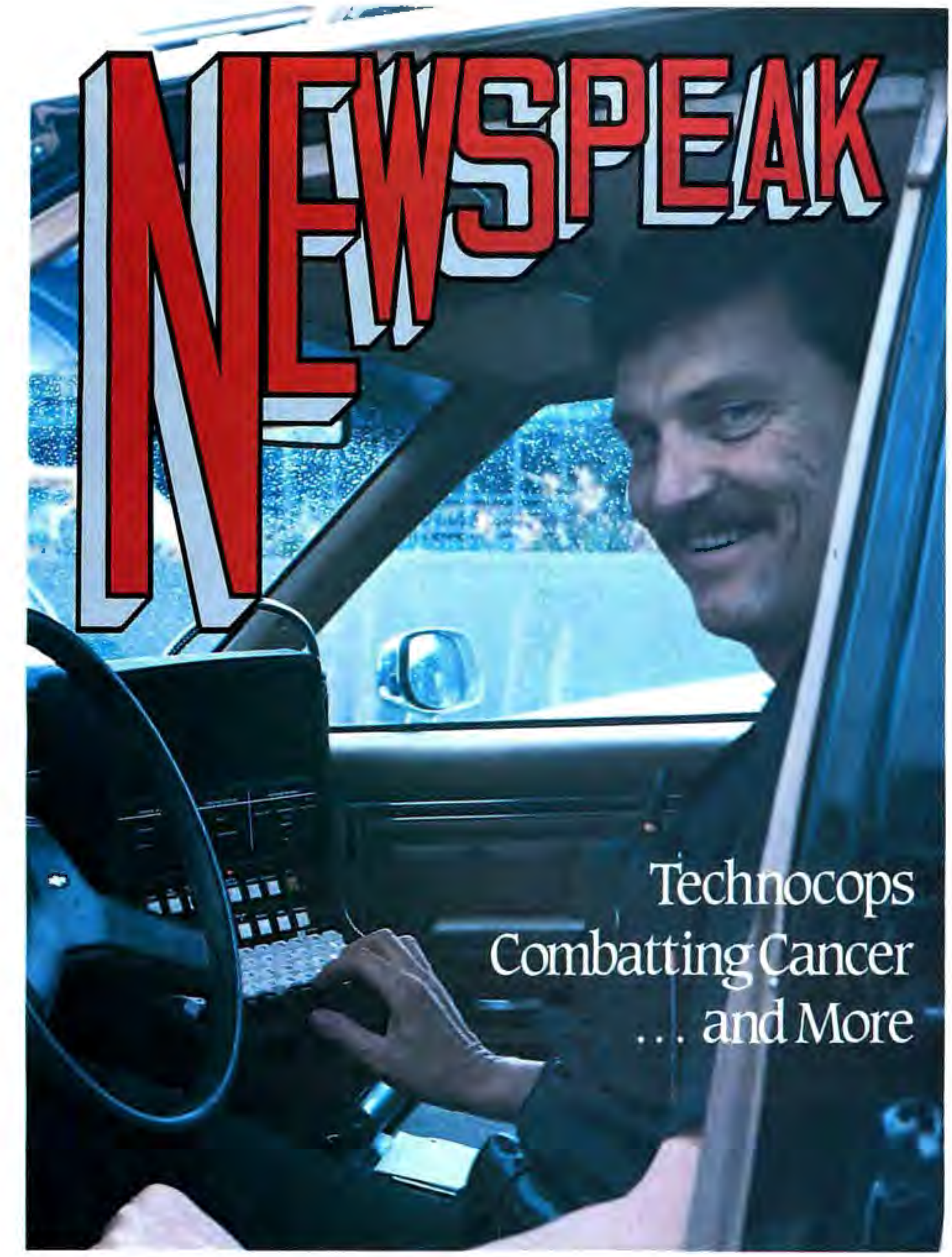
With all that, you still don't have to learn commands or pay a lot. BATCHPRINT is just $75.00.

PYXEL APPLICATIONS

2917 Mohawk Drive
Richmond, VA 23235
(804) 320-5573

PYXEL Visuals - $98
BATCHPRINT - $75
MasterCard/Visa Accepted

Requires: IBM PC with PC-DOS; 128K RAM; Color/Graphics Adapter; Graphics Monitor; EPSON MX-100, MX-80 with GRAFTRAX, FX-80 or IBM Graphics Printer. DOS 2.0 and XT compatible.



LAW ENFORCEMENT AGENCIES ENTER THE COMPUTER AGE

The law enforcement officer punches in his badge number on his car's computer terminal and receives orders to proceed to the neighborhood liquor store. He races to the store, where he enters and spots three innocent-looking customers. No more than fifteen seconds elapse before the trouble begins. A man takes a gun from his vest and threatens the cashier. The policeman now must make his move.

The suspect whirls about and points the gun at the officer. Without warning the suspect fires; the officer clutches his chest but doesn't fall to the ground.

The scene takes place in no ordinary liquor store. It's a simulated crime—one of many regularly enacted at Laser Village, the Los Angeles Sheriff's Department's new $514,000 training facility. Laser Village consists of a one-bedroom house and a two-story commercial center with a liquor store, gun shop, business office, bank, doctor's office, and a saloon—all realistically furnished. This environment is the setting for confrontations between trainees clad in laser-sensitive vests—which, when struck by a laser beam, react by chirping and blinking—and play-acting suspects armed with special laser-equipped revolvers and shotguns. The

GOTO page 179, column 1

CRAY CRUNCHES THE COMPETITION IN CHESS TOURNEY

Who has the best chess-playing computer program in the world?

Every three years, the Association for Computing Machinery makes use of its annual convocation to determine just that. The latest decision was reached at the Sheraton Center Hotel, New York, the weekend of October twenty-second last year. The occasion was the fourth international gathering of electronic chess champs and hopefuls for the World Computer Chess Tournament. Twenty-two teams, representing the United States, Canada, Germany, Holland, Finlaud, Sweden, and Austria, came to play.

Ken Thompson's *Belle*, from AT&T's Bell Laboratories, the reigning world champion, had trounced Bob Hyatt's *Cray Blitz* in the Twelfth North American Championship in 1981. Both programs had been playing at master levels for the last two years. Was *Cray Blitz* looking for a rematch at the world meet?

It got it. The spoiler for *Belle*, as it has been at previous meets, was *Nuchess*. This artificial intelligence-styled program, which examines possible plays by a series of selective criteria rather than looking for every possible move in a series, took on *Belle* in the third round and handed it a loss, *Belle*'s first ever in four years. *Belle*'s point spread never recovered. Hyatt's *Blitz*, running on a dizzyingly fast dual-processor Cray computer, played to a draw with *Nuchess* in the fourth round. *Cray Blitz* went on to win the tournament in the fifth round.

"Hyatt's program was already very tough," comments tournament organizer Monroe Newborn, "and the speed was enough to make the difference."

Some observers at the tournament said that *Belle*—on a portable machine—may have lost because of an "electronic concussion." *Belle* was in a car accident shortly before the tournament.

Tony Scherzer's *Bebe* snagged second place by defeating *Nuchess* in the final round. Scherzer was one of several programmers present whose computers played much better chess than their creators. These programmers found it hard to tell whether a puzzling computer move was a subtle piece of strategy or a programming error. Other programmers who happened to be more skilled players themselves watched in horror and

GOTO page 182, column 2

Simulating Path of Particle Beam Bombardment

Computers Help Doctors Fight Eye Cancer

At one time the treatment for a tumor located within the eye involved either radical surgery or dangerous radiation. Neither strategy was risk-free. When surgery was done, the entire eyeball usually had to be removed to prevent the cancerous cells from infecting other body sites. And because the gamma rays used to irradiate the tumor are unfocusable and nonuniform, radiation treatment often meant the destruction of vision. In addition, tumor regrowth or other radiation-connected complications occurred in 50 percent of the cases.

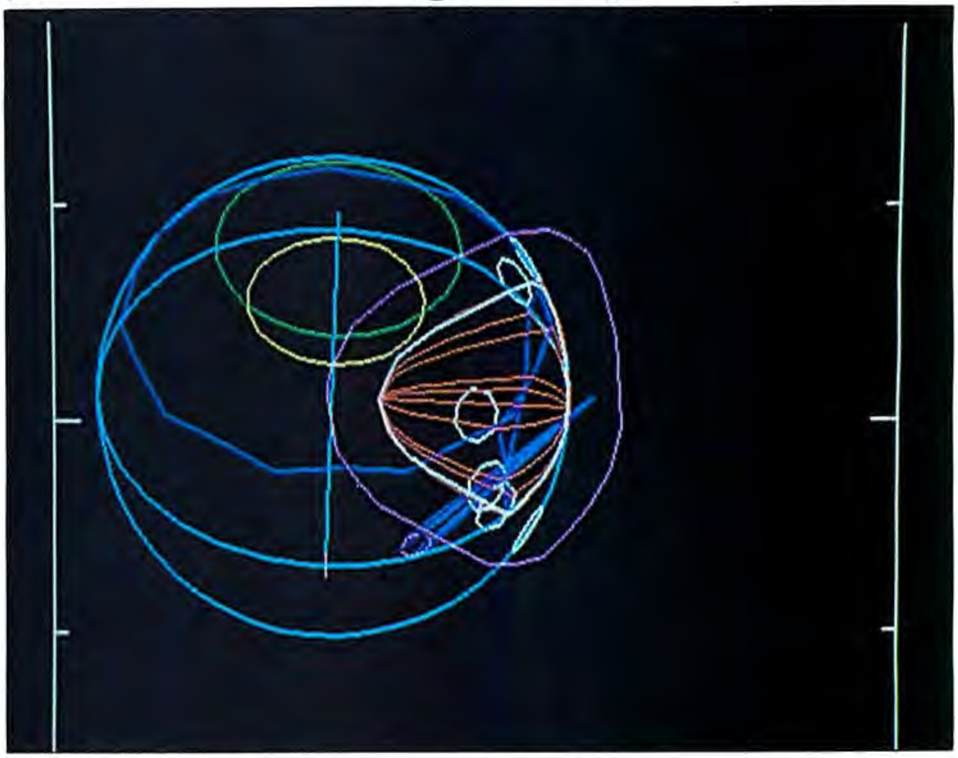
A program created by Dr. Michael Goitien of Massachusetts General Hospital now offers people afflicted with eye tumors a better chance of retaining their vision. Written in Fortran to run on a Digital VAX 11/780 host system, the program aids physicians and physicists by providing a computer model, or graphic interpretation, of a patient's eye. By interacting with the program, the physician can determine the appropriate particle course to attack the tumor, as well as the specific dosage of radiation required to accomplish the task.

A modified version of Goitien's program is currently being tested at Lawrence Berkeley Labs in California by a team of doctors and computer scientists. As it turned out, the Berkeley-based research team found themselves using a different set of graphic monitors and heavier particles (helium ions) for tumor bombardment, so LBL's Dr. Samuel Pitluck rewrote I/O routines and other algorithms to produce a program tailored to these specific conditions.

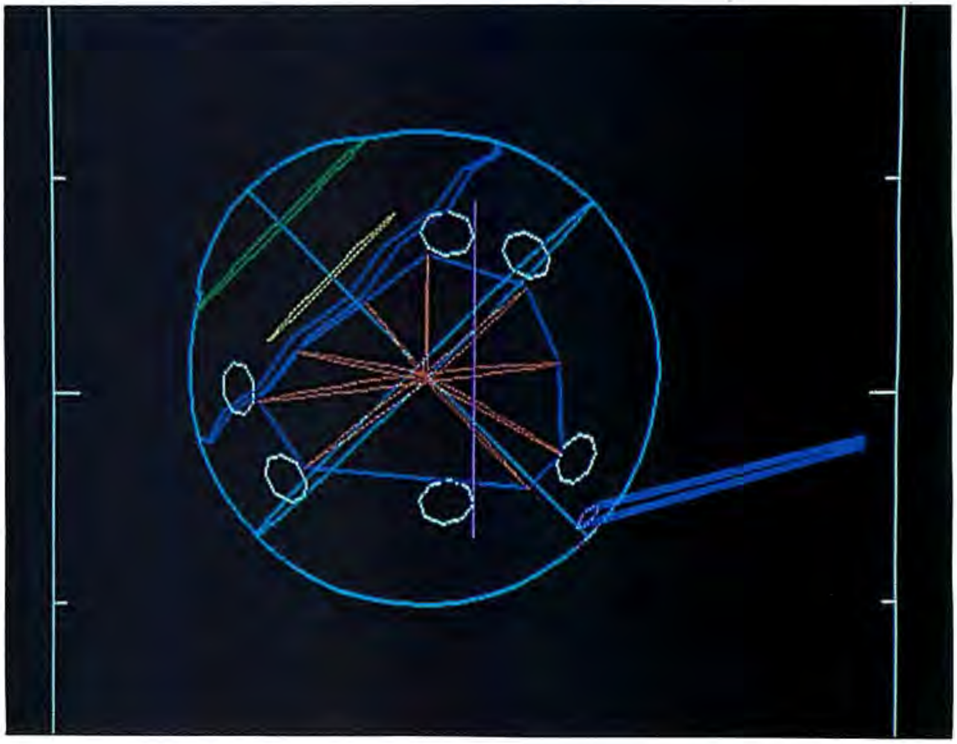
The helium nuclei needed to bombard a tumor are produced by LBL's 184-inch cyclotron. An alternating electrical field accelerates helium atoms outward in a spiral and at right angles to a fixed magnetic plane. The cyclotron strips electrons from the atoms, creating alpha particles. Because they are electrically charged (unlike gamma rays), these particles can be focused and guided and can destroy a tumor without damaging the eye. Instead of delivering energy to all the tissue encountered, as x-rays do, these ions release their energy at the end of their path—at the tumor itself.

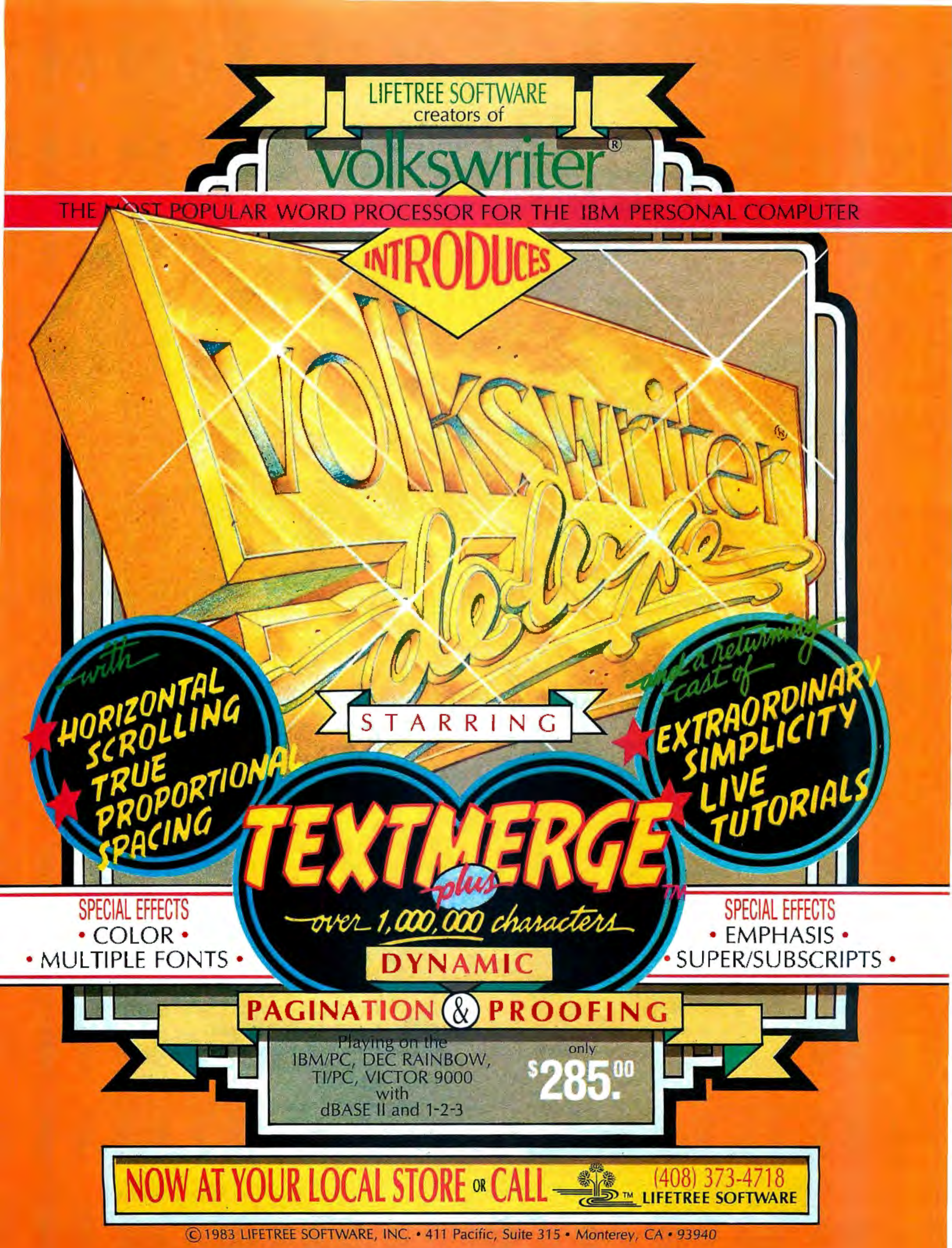
Normally, a physician can spot a tumor by looking directly into the patient's eye. Once the tumor has been located, ultrasound and wide-angle photography are used to create an image with depth, from which scale drawings of the tumor can be made. Circlets, or rings, made of the metal tantalum, are

GOTO page 180, column 2



Above, an image of the eye, generated by the computer, from the point of view of the projected ion beam. Red outlines the tumor itself, with the five white circles representing five of six tantalum rings. The yellow and green ovals represent the eye's lens and cornea respectively. The optic disk and nerve, which connect the retina to the brain and allow you to see, are in the area designated by the pink line. These critical structures are located at the rear of one's eye. Below, lateral image of the eye. The actual spread of the tumor and six tantalum rings are readily seen. With these graphic simulations, doctors can easily plot the best path for the beam of helium ions used to bombard the tumorous area.




LIFETREE SOFTWARE
creators of
volkswriter®
THE MOST POPULAR WORD PROCESSOR FOR THE IBM PERSONAL COMPUTER
INTRODUCES
volkswriter® deluxe
with
HORIZONTAL SCROLLING
TRUE PROPORTIONAL SPACING
STARRING
and a returning cast of
EXTRAORDINARY SIMPLICITY
LIVE TUTORIALS
TEXTMERGE™
plus
over 1,000,000 characters
SPECIAL EFFECTS
• COLOR •
• MULTIPLE FONTS •
SPECIAL EFFECTS
• EMPHASIS •
• SUPER/SUBSCRIPTS •
DYNAMIC
PAGINATION & PROOFING
Playing on the
IBM/PC, DEC RAINBOW,
TI/PC, VICTOR 9000
with
dBASE II and 1-2-3
only
$285.00
NOW AT YOUR LOCAL STORE OR CALL
(408) 373-4718
LIFETREE SOFTWARE

The installing of portable terminals in LAPD squad cars began earlier this year. So far both cops and those they serve have benefited from the improved communications possible with the $42-million Emergency Command Control System (ECCS).

Law Enforcement

continued from page 175

beams, which travel as far as sixty feet, show precisely where real bullets would hit. Previous training methods, which involved play-acting suspects and trainees using guns that fired blanks, were not nearly as accurate as the Laser Village approach.

The program's goal is to teach deputies the proper time to use guns and to make them more aware of the "real" consequences of their actions. The village and the equipment (Laser Village has thirty revolvers and four twelve-gauge shotguns) do not, surprisingly, impose a taxpayer's burden, since everything is financed through private foundations.

Whether in training or in actually enforcing the law on the streets, police in general are increasing their use of modern technology. Improving communications through the use of computer technology and controlling suspects through the use of nonlethal weapons, such as the Taser gun, are some examples of this trend.

Efforts to use technology are motivated, in part, by the fact that staff positions have been eliminated because of budget cuts. (There are 700 fewer positions in the Los Angeles Police Department than there were in 1975.) The police must also keep pace with criminals, who themselves have access to sophisticated technologies. For both reasons, the effectiveness of the individual law enforcement agent must improve. The age of the technocop has arrived.

The LAPD's Emergency Command Control System (ECCS, or X for short) was implemented throughout Los Angeles last year and has resulted in the replacement of squad car radios with computer terminals. The $42-million system enables officers to tap into computer data banks on a local, state, or nationwide level. When requesting information on wanted persons or stolen vehicles and property, an officer uses a portable keyboard to send a message to a central computer that automatically processes the request, eliminating the need for an operator to handle incoming calls. Likewise, requests for help received at the complaint center are sent immediately to the officer's portable ECCS terminal.

Training in the use of the system requires eight hours of classroom instruction and eight hours on an operational terminal. While no formal study of the effectiveness of the ECCS has yet been made, its impact has already been felt in the San Fernando Valley, where the system was originally implemented this past spring. The use of ECCS has already led to an increase in the recovery of stolen cars.

As well as improving communication systems, the police in Los Angeles and other cities in the nation are experimenting with advanced weaponry. The Taser, an electronic dart gun employed by the LAPD, is "not a replacement for a lethal weapon," according to Deputy Chief Clyde L. Cronkhite. Rather, it is an instrument designed to suppress uncontrollable suspects painlessly. A dart has a wire attached to it, and, when the suspect is hit with the dart, the officer pulls the gun's trigger, causing the dart to emit a low-amp, high-voltage charge that produces temporary paralysis. Once the trigger is released, the shock ceases. To be effective, the darts must be fired at a maximum distance of twenty feet from the suspect.

Each month, experts in law enforcement and related fields convene at the Security Pacific National Bank headquarters in downtown Los Angeles for a meeting of Forum 2000 to discuss future technology and its effects on law enforcement. Among the topics discussed at Forum 2000: an automated fingerprint identification system, in which a person's index fingerprint would be digitally stored and transmitted (this could be especially useful to retail stores); the use of 3-D photography and computers to produce lifelike portraits; the practice of freezing criminals for the duration of their prison terms.

Other ideas include police cars capable of

GOTO page 180, column 2

Law Enforcement

continued from page 179

deactivating the ignitions of other cars and the use of satellite technology for communications and keeping track of criminals. A low-power microwave transmitter that could be implanted in a convicted criminal is another possibility. The transmitter would send a signal that could be picked up by a series of satellites capable of pinpointing the exact location of the transmitter so that the whereabouts of the criminal could always be determined.

After ideas are discussed by the one-hundred-member panel, those deemed worth investigating are presented to the planning and research division of the LAPD. This group elaborates on the ideas and puts them into project form. Besides attending the monthly meetings, Forum 2000 participants are active in subcommittees—groups that explore such issues as management/employee relations, crime-related issues, future funding of police activities, public perceptions of the police, and technological and demographic changes.

As the use of technology in law enforcement agencies increases, changing the way police officers do their jobs, Cronkhite believes that the image of the police will also undergo a change.

"We think that technology will not make a cold, dispassionate police force but rather will give the officers more time to be of personal service, to occupy themselves less with the mundane tasks technology can perform for them."

The technocops have arrived. Time, money, and society's response will determine whether they are here to stay. JG

Eye Cancer

continued from page 176

then sewn into the eyeball's exterior to mark the dimensions of the tumor. The rings are needed because the eye cannot be imaged by x-ray; the rings act as markers that map the shape and location of the tumor.

At this stage, the computer comes into play. Dr. George Chen, a physicist, and Dr. William Saunders, assistant professor in the University of California at San Francisco's Radiation Oncology Department, manage the computer modeling of the patient's tumorous eye. Chen and Saunders make certain that the final direction of the particle beam and its depth within the tissue are optimized to prevent damage to significant structures within the eye itself, including the macula lutea (near the retina's center, where visual perception is most accurate), the optic disk and optic nerve, and the lens.

The trick to this new technique lies in obtaining the correct "gaze angle" for the patient, who must look at a small flashing red light. This angle separates the eye structures and allows for a clean shot at the tumor by the helium particles. By controlling the gaze angles, Chen and Saunders can visually determine the best path for the beam.

"The eye can be looked at laterally, from overhead, or from any one of many different angles," states Chen. A two-and-a-half-millimeter blue line encircling the envelope of the red tumor is automatically calculated and displayed on the screen. If this line intersects with, or is too close to, a significant structure, then the computed gaze angle is incorrect. Reentering data, the physicians can rotate the eye model until an effective angle that avoids proximity to critical structures is found.

Depth of radiation penetration is also calculated by the computer, as are the actual doses that will be played over the tumor. "Our big advantage," Saunders says, "is that we can stop this radiation instead of having it whistle on through the tumor to other areas of the eye. We can also make sure that the beam sweeps back and forth correctly to give the tumor a uniform dose of radiation."

Since radiation of the eye's lens can cause cataracts, an effort is made to avoid this. (Fortunately, however, cataract surgery is now an outpatient affair and easily managed.)

Once the computer calculations have been made, a plaster-of-Paris mask conforming to the patient's head is constructed. This model is then vacuum-formed using polystyrene, with openings for the eye. The mask makes it easier for the patient to hold still. It takes forty or fifty minutes to position the patient correctly for the beam penetration, which only takes a minute or so.

The largest tumor successfully treated to date measured eighteen millimeters. Of the 130 patients treated so far, 90 percent have been treated successfully—with success defined as retention of the eye for up to six years, tumor control, and no metastasis (transfer of a disease-producing agency from the original site to another part of the body). Similar research is under way regarding treatment for victims of certain spinal tumors that have proven to be resistant to x-ray treatment. HL

RETURNABLE SOFTWARE FOR THE IBM-PC®

Eliminate the risk of buying software that doesn't meet your needs.

With Computer Inventory Control, you eliminate the risk of buying software that doesn't meet your needs. Buy any of the most popular IBM-PC software programs at our special price. If it doesn't meet your needs, return it within 15 days for a refund of your purchase price less a restocking fee of only 15% of the manufacturer's list price.

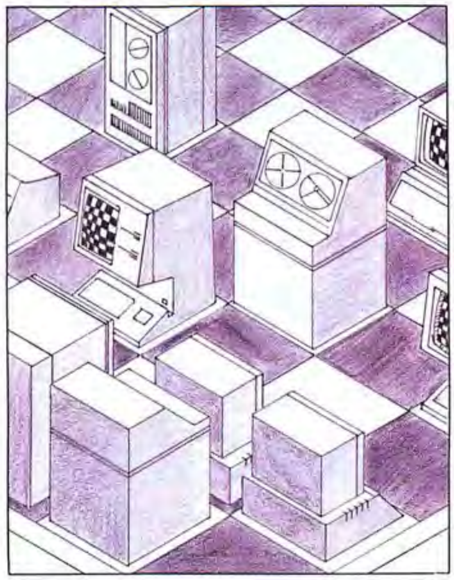
To place your order call (800) 824-7888 Operator 320
For more information and our FREE catalog call (412) 687-2000

COMPUTER INVENTORY CONTROL, INC.

159 Techview Terrace • Pittsburgh, PA 15213 • (412) 687-2000

WE HONOR MasterCard

VISA



Chess Tourney

continued from page 175

frustration when their programs blundered into fatal errors or threw away obvious opportunities.

In a special ceremony at the start of the tournament, before *Belle* got beaten by *Nuchess* and *Cray Blitz*, the U.S. Chess Federation officially awarded *Belle* the rank of master; this was the first time a computer program had been so recognized. The title, however, did not pass to *Cray Blitz* as simply as would, say, the championship in a heavyweight title bout. To get the master rating, "you have to play twenty-four rated games," explains Bob Hyatt. His program had played nine or ten games prior to participating in this five-game tournament.

Hyatt attributes his success to *Blitz*'s increased depth of search in looking for possible moves from current positions. "Up until 1983, *Belle* was the only computer capable of eight-ply searches [four full moves by each side, plus checks and captures]. It was looking at least eight half moves deep, and a lot deeper in some circumstances. We went to work on our program: I rewrote all the stuff for the multiprocessor; Harry Nelson at the Lawrence Livermore Laboratory optimized it for the Cray, and we found ourselves going eight ply.

"Search depth and horsepower is the name of the game, and we happened to have an awful lot of horsepower this year."

The program and its proud parent will not be resting, however. Hyatt is hoping to have a four-processor version of *Cray Blitz* running by this summer—a program that will think twice as fast as the current model. His champ has a full schedule until then. Chess master and programmer David Levy has long stated that he can beat any computer, and *Omni* magazine has agreed to pay $5,000 to anybody with a computer that can prove him wrong. Hyatt wants to take that bet and is looking for a sponsor for a Las Vegas showdown in January. In February, *Cray Blitz* will be in France, at the invitation of the French Ministry of Culture, for a three-day exhibition game. This summer, Carnegie-Mellon University will pit all the top-rated programs against equally rated humans.

For the moment, whatever the future may hold for Hyatt's brainchild, the fast-moving, highly competitive world of computer chess has a new champ. "We were all very pleased at the outcome. Cray Research is happy because they're number one for the next three years," says Hyatt. "When we get the new version finished, it'll be able to look at forty to fifty thousand positions a second. We're going to make hay while the sun shines."AC

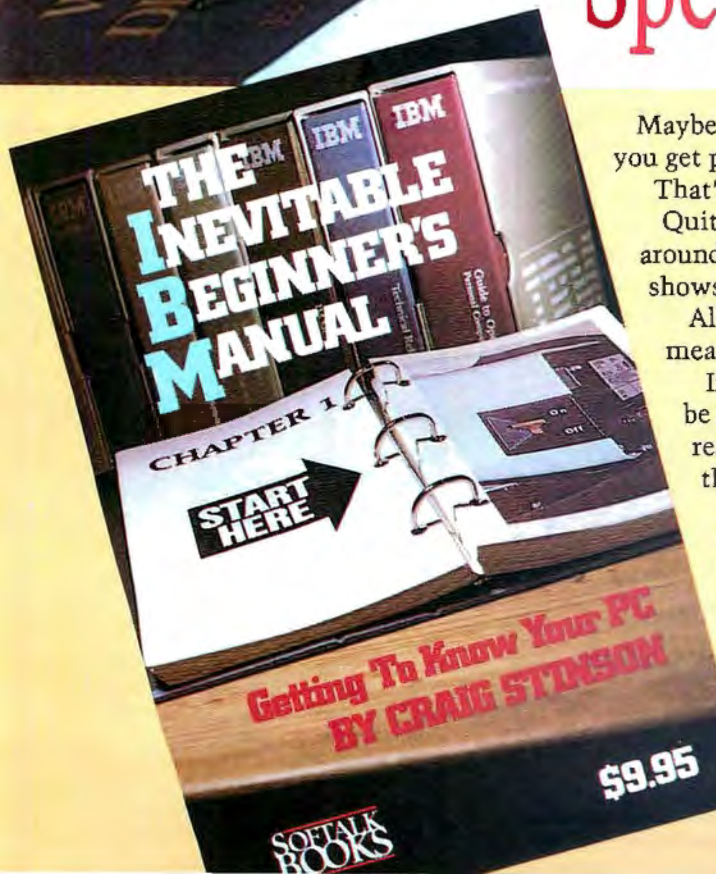
NEWSBITS

Δ **Far East Computing.** The Interface Group (Needham, MA) has announced that it will hold a Comdex in Japan next spring. The three-day computer dealer show will take place at the Harumi exhibition facility in the Tokyo International Trade Center March 26–28, 1985. Closer to home, and coming up sooner on the calendar, is the premiere of Comdex/Winter at the Los Angeles Convention Center April 5–7, 1984. Also, the fourth annual Comdex/Spring will be held in Atlanta May 22–25.

Δ **Spring ME Festival.** Well, there probably won't be any US Festivals in 1984, but the ME Festival is returning. Sponsored by California State University, San Bernardino, and various organizations, including the U.S. Small Business Administration, the ME (Microtechnology for Everybody) Festival will be held April 27–28, 1984, at California State University, San Bernardino. Besides giving the surrounding community a look at advanced technology, this year's ME Festival will feature the Robot Olympics. Schoolchildren from kindergarten to high school will program robots for competition in "compulsory figure" and "free-style" events. For more information on the ME Festival and the Robot Olympics, contact the ME Festival coordinator at CSUSB's Computer Center.

Δ **Market Manual.** Writer's Digest Books (Cincinnati, OH), a publisher best known for its how-to books and market directories for

Is
Your
PC
Speaking in Tongues?



IBM
THE INEVITABLE BEGINNER'S MANUAL
CHAPTER 1
START HERE
Getting To Know Your PC
BY CRAIG STINSON
$9.95
SOFTALK BOOKS

writers, artists, and photographers, has recently published the 1984 *Programmer's Market*. Edited by Brad M. McGehee, the book contains several chapters on how to put together and market programs, and numerous market listings, including both software publishers and magazines that buy programs from free-lancers. Writer's Digest Books is currently putting the finishing touches on a companion volume, *The Complete Guide to Writing Software Users' Manuals*, scheduled for publication this spring.

Δ **The Way We Are.** The Fourth Annual Talmis Conference takes place in Chicago February 15–17, 1984. The conference typically attracts hundreds of executives responsible for strategic planning, marketing, and development of home and educational computer industry products. The conference includes many opportunities for attendees to discuss computer industry issues and trends. For more information, contact Talmis at its Oak Park, Illinois, office.

Δ **Rotate 'n' Roll Robots.** Virtual Devices (Bethesda, MD) is sponsoring the First Annual Robot Roll-Off. Participants have to build a robot that can maneuver a simple course on its own, open doors, detect sound, light, and motion, and do it all against the clock. Robot Roll-Offs are scheduled in cities across the country during the summer of 1984. The Roll-Off grand prize winner receives $5,000, while runners-up get cash and merchandise. Participants can enter their own creations or existing robot kits, such as Heathkit's Hero, RB Robot's RB5X, and others.

Δ **Keeping an Eye on the Books.** Howard W. Sams and Company (Indianapolis, IN) has released a new book—*Microprocessor-Based Robotics*—for robot hobbyists. The 224-page book covers ways to build robot hands, legs, and bodies using common household items. Written by Mark J. Robillard, the book also discusses how to make robots see, touch, and move, and how to use microcomputers to control robot movement. Also new from Sams is a book about satellite communications, called *The Birds of Babel*, and a listing of terms used in the security industry, called the *Security Dictionary*. Written by Hal Glazer, *The Birds of Babel* covers the basics of satellite technology, legal issues, social and political concerns, and the business side of the industry. *Security Dictionary*, compiled by Richard A. Hofmeister and David J. Prince, covers video equipment, computer hardware and software, ultrasonics, infrared sensors, microwaves, and how such things apply to the security field. The book also includes tables of security and fire protection symbols.

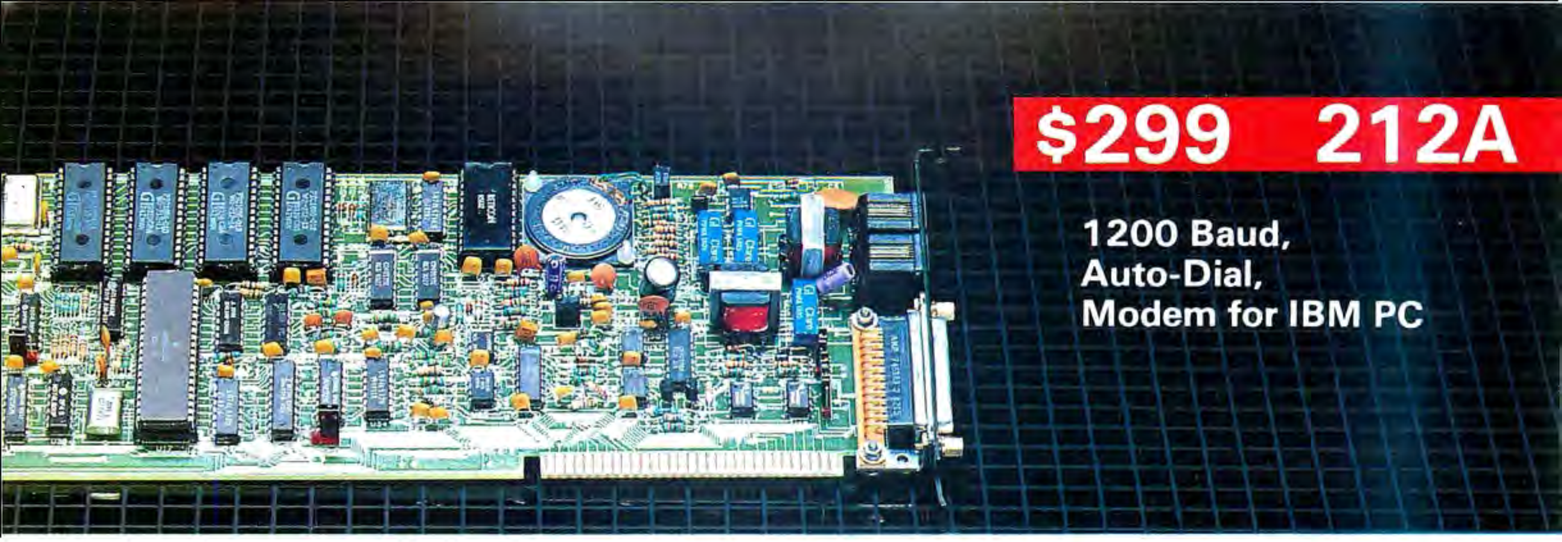
Δ **The Sound of Computing.** The IEEE 1984 International Conference on Acoustics, Speech, and Signal Processing will be held March 19–21, 1984, at the Sheraton Harbor Island Hotel in San Diego, California. For more information, contact Sam Vilione of Interstate Electronics in Anaheim, California.

Δ **Computing Albion.** Britain is lagging behind the U.S. in its use of microcomputers, according to San Francisco, California-based computer services firm Ferrin. Company chairman David Ferris, who spoke recently at an industry gathering in London, says there are similarities between what users are doing in the U.S. and what they are doing in the U.K., "but there seems to be a time lag of about a year between the two." Much of the software being used in the U.K. is indigenous—for instance, data management packages such as *DMS* and *Cardbox*—although U.S. software products still abound. Ferrin puts the size of the British personal computer market at about one-tenth the size of the American market. According to Ferris, many Brits look on their lagging position as a blessing in disguise. "The lag means it's possible to benefit from the experience gained and lessons learned in the U.S."

Δ **Hi-Tech Highway Help.** Starting late last year, the Automobile Club began making TDD (telecommunications device for the deaf) units a part of its Emergency Road Service. TDDs are about the size of an ordinary typewriter and allow speech- and hearing-impaired people to send typewritten messages over telephone lines. Currently, some telephone companies distribute portable TDD units; speech- and hearing-impaired individuals can apply for a TDD through their local phone company. Those hearing-impaired Auto Club members who own a TDD can request emergency road service when their vehicle breaks down, if they can get to a pay or private telephone. Freeway call boxes are not yet equipped to handle the TDD equipment.

Δ **Captive Penguins in a Totalitarian State?** Everything you'll ever want to know about penguins and then some is likely to be on a new computer system at Hubbs-Sea World Research Institute. The Atlantic Richfield Foundation has donated $20,000 to finance a system that will help collect and analyze growth, breeding success, health, and behavioral characteristics of each of the several hundred penguins in Sea World's Penguin Encounter in San Diego. Big Penguin is watching you! ▲

Editor **David Hunter**
Contributors **Andrew Christie, Jane Greenstein, and Hartley Lesser**



PC 212A/1200:

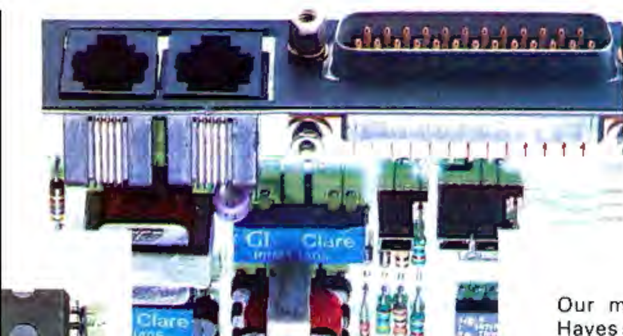
A GENUINE BREAKTHROUGH

NO CORNERS CUT

We included every feature you would want in a modem card. It's FCC registered for direct connection to your modular phone jack with the cord which is included. There is a separate modular jack for your telephone or you can listen through the onboard speaker. The auto-dialer works on rotary lines, tone lines, or a combination of both, and will pause for use with Sprint or MCI. It will work in originate or auto-answer modes. A separate microprocessor, a Z8, controls all the functions.

ADVANTAGE #2

Optional external serial port connector



AN ASYCHRONOUS ADVANTAGE

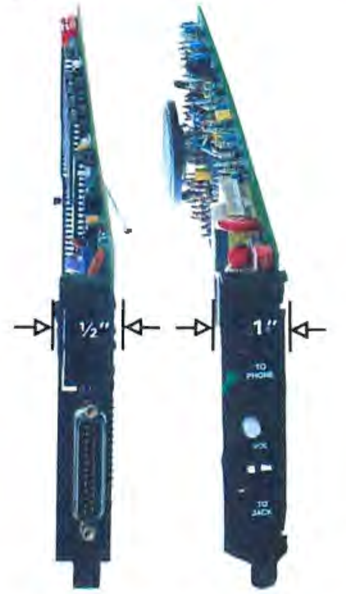
The modem board is addressed in the software as COM1 or COM2 and we have a handy little option you ought to consider. If you would like to use the asynchronous communications port when your modem is not in use, we will add a connector and the necessary circuitry for just $20. This saves you the hundred bucks or so you would spend for another async card and saves a valuable slot. It can be configured as COM1 or COM2 and works just like IBM's does.

THIN IS IN

It plugs into your IBM PC or XT and occupies any one slot since it is just 1/2 of an inch thick. This is made possible by using a special speaker which is just 1/8" tall. Competing brands either use a convential cone type speaker, or they just skip the speaker altogether. Some modems also have large transformers which allow rob valuable space.

ADVANTAGE #3

Just one slot in PC or XT



LET'S TALK SOFTWARE

Our modem is 100% compatible with the Hayes software commands so you can use any of the popular communications packages like IBM's Asynchronous Communications Support, CrossTalk, Transend, or PC Modem. We go one better than the competition. We **include** PC-TALK III. PC WORLD magazine referred to it as "the benchmark that other PC communications packages are measured against." It stores phone numbers, handles setting the modems characteristics, saves to disk files, transmits from disk files, even binary files. You can program up to forty keys to have things like passwords and log-on information be entered when you hit them. And to make sure data is sent and received accurately, the XMODEM protocol detects errors caused by poor line quality and automatically retransmits the data.

WHY BUY FROM US

Because besides having the best product on the market, we stand behind it and you. You get factory direct technical support after the sale. If an any time during the one year warranty period your modem should require service, we will fix or replace it within 48 hours.

Notice also there are no hidden charges in our price. Nothing extra for credit cards or COD charges. We even pay UPS shipping. If you still are not convinced, and are ready to buy another brand of modem, ask them if they will take our acid test.

THE ACID TEST

Qubie gives you a 30 day satisfaction guarantee on your modem. If you are not completely satisfied we will refund the entire amount of your purchase including the postage to return it. If you can, get anyone selling one of our competitor's products to give you the same guarantee. Buy any modem you like and return the one you don't like. We know which one you will keep.

ORDER TODAY

It's easy to order by mail or by phone.

BY PHONE: Call us and one of our sales staff can answer any questions you have and take your order. Have your Visa or Mastercard number handy when you call.

(805) 987-9741

BY MAIL: We need your name and street address, daytime phone number, how many modems you want, and whether your computer has single or double sided drives.

VISA & MASTERCARD

$299 includes: PC 212A/1200 auto-dial modem card, PC-TALK III software, cord to connect to modular phone jack, and manual. 1 year limited warranty.

VISA master charge

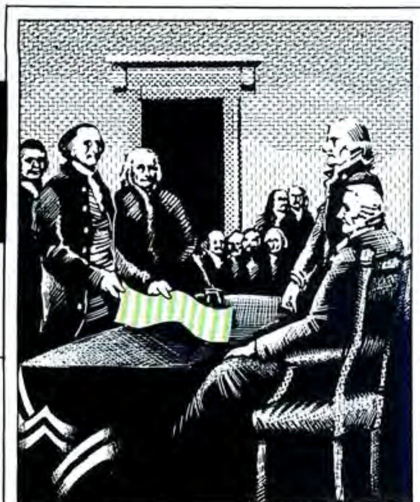
Optional: Connector and circuitry to use serial port for another serial device $20.

SHIPMENT

We pay UPS surface charges. UPS 2 day air service add $5 extra. Credit card or bank check orders shipped next day. (Personal checks take 18 days to clear)

QUBIE'

4809 Calle Alto
Camarillo, CA 93010



THE RIGHT TO ASSEMBLE

by Ray Duncan

The CLEAN Utility

This month we present a new utility program that illustrates some of the powerful new functions provided for the assembly language programmer under DOS 2.0. The CLEAN program will fix up a text file created by nearly any word processing package so it can be read by other screen editors or line editors. It does this by stripping off the high bit of each byte, expanding tab characters to the appropriate number of space codes, and throwing away all other control characters except for carriage return, line feed, and form feed. An end-of-file marker (1AH) is always put at the end of the file. CLEAN is especially useful for massaging *WordStar* document files into a form that ordinary program editors can cope with.

CLEAN illustrates the use of the DOS 2.0 "extended" disk functions, which can accept pathnames in file specifications and perform operations on records of any size at any byte offset in a file. These functions differ radically from the set of file and record operations present in DOS 1.1 (which were originally designed to be as compatible as possible with CP/M-80); they show a clear tendency toward providing a Unix-like environment.

How To Use CLEAN. The file Clean.exe must either be on your current disk drive in the current subdirectory or in one of the subdirectories specified in the most recent *path* command. Assuming this is the case, you simply enter:

A)CLEAN filespec (return)

The filename can include a disk drive and a full path specification. For example, you could enter:

A)CLEAN \ SUBDIR \ MYFILE.DAT

The file created as a result of CLEAN's labors is always given the same filename as the input file, but its extension is forced to .cln. The new file is stored in the same subdirectory as the original data file, *not* in the subdirectory that holds the file Clean.exe. The contents of the original file are unchanged.

In the previous example, when CLEAN finishes running, the file Myfile.dat is unmodified, but a new file Myfile.cln has been created in the same subdirectory; the new file holds the cleaned-up and stripped-down text file.

Program Outline. The general flow of the program CLEAN is as follows:

1. Push the contents of DS and an offset of 0 onto the stack—the combination of these is the thirty-two-bit address that will be used for a final return to DOS when the program finishes.

2. Look at the command tail that's left by DOS at offset 80H in the Program Segment Prefix to see if the operator has specified a filename. If none is specified, print an error message and exit.

3. If a filename is found, transfer it to a special buffer and terminate it with a 0 byte. Place a second copy of the filename with the extension .cln in a second buffer for use as the output file specification.

4. Attempt to open the specified input file. If unsuccessful, terminate with an error message. Possible reasons for an inability to open the input file are: The file doesn't exist at all, the file doesn't exist in the current subdirectory or in the subdirectory specified by the path in the file specification, and the path given by the operator does not exist.

5. Attempt to create a new file for output, with the same filename and an extension of .cln. If a file already exists under that name, truncate it to zero length in preparation for writing. This operation fails only when there's no more room in the disk directory.

6. Fetch a character from the input file. If the character is an end-of-file (EOF) marker (1AH), go to 10.

7. If the character from the input file is not a control code, write it to the output file; increment the column counter and go to 6.

8. If the input character is a control code and it is a carriage return, line feed, or form feed, set the column counter to 0; write the character to the output file and go to 6.

9. If the control code is not a tab code, discard it and go to 6. If it is a tab, divide the column counter by 8 and use 8 minus the remainder as the number of spaces to send to the output file; then update the column counter. This expands the tab. The tab code itself is discarded. Go to 6.

10. (Control comes here when an end-of-file marker is found.) Write the EOF marker to the output file, then close both files. Display the message "File Processing Completed" and return to DOS.

Detailed Program Description. Since CLEAN is to be assembled and linked into an exe file, it contains three segments. A sixty-four-byte stack segment is declared in lines 310 through 315; this is used as a scratch area by PUSH, POP, and CALL operations. A data segment containing the program's variables, constants, and operator messages is declared in lines 249 through 307. We make it addressable by putting the address of the beginning of the data area into DS and ES; this is done in lines 35 and 36 and 39 and 40.

The executable machine code is contained in a code segment named CSEG, which is declared in lines 25 through 246. The beginning of the code segment is followed by an ASSUME statement (line 27), which tells the assembler which segment registers you will use to address the various segments of your program.

Within CSEG, there is one main routine called CLEAN and a number of subroutines. The procedure CLEAN receives control from DOS and performs the final exit back to DOS, so it is declared as a FAR procedure. The END statement in line 317 includes the name CLEAN to define the entry point of the program for the linker and loader.

There are quite a number of short subroutines; each performs a simple and distinct function when called by the main routine CLEAN.

The subroutine INFILE scans the command tail for the name of the file to be used as input, chops off leading and trailing spaces and other spurious codes, and transfers the name to an internal buffer. The procedure OUTFILE takes that filename, copies it to another buffer, and changes the extension to CLN.

The routines OPEN_INPUT and OPEN_OUTPUT open and create the input and output files respectively, using DOS functions 3DH and 3CH. The address of the path and filename string is passed to DOS, and a sixteen-bit value, or token, is returned. This token is called a *handle* and is used to request read and write operations on the file. Note that the access type (read only, write only, or read/write) also is passed to DOS in the open or make request. If you attempt to open a file for writing that is marked with the read-only attribute bit in the directory, an error condition will result.

The subroutine GET_CHAR reads a one-byte record from the input file and returns it to the caller in register AL. If the physical end of the file is reached without an end-of-file marker having been read (some editors don't use one), GET_CHAR senses this and supplies one to the main routine. DOS function 3FH is used; it requires a buffer address, record length, and handle as arguments.

The procedure PUT_CHAR writes the character passed in register AL to the output file. PUT_CHAR uses DOS function 40H, which requires a buffer address, record length, and handle as input parameters. If the disk is full, an error code is returned to the main program.

Finally, the routines CLOSE_INPUT and CLOSE_OUTPUT request a close operation (DOS function 3EH) on the input and output files respectively. Actually, closing the input file is not necessary in this case, but it is a good habit to get into. The output file must be closed properly so that DOS will update the information in its directory entry (length of the file, date and time last modified).

After looking over the program on page 188, read pages D-35 to D-37 of the DOS 2.0 manual for descriptions of the functions you've used. The error codes returned are described on pages D-14 and D-15.

For the purposes of this column, CLEAN was written so that it performs one-byte record I/O—which makes the logic of the program much clearer. It also makes the program very slow because of the overhead of performing two calls to DOS for every byte transferred from one file to the other. To speed this program up dramatically and test your understanding of its logic, modify PUT_CHAR and GET_CHAR so that they request 1,024-byte record transfers from DOS and then internally block/deblock the records into bytes.

→

```
                              name    clean
                              page    55,132
                              title   'CLEAN—Filter text file'

                      ;
                      ; CLEAN—a utility to filter text files.
                      ; This program removes all control codes except
                      ; for line feeds, carriage returns, and form
                      ; feeds, strips off the high bit of all characters,
                      ; and expands tabs. Can be used to make a WordStar
                      ; file acceptable for other screen or line editors,
                      ; and vice versa.
                      ;
                      ;

 = 000D               cr      equ     0dh             ;ASCII carriage return
 = 000A               lf      equ     0ah             ;ASCII line feed
 = 000C               ff      equ     0ch             ;ASCII form feed
 = 001A               eof     equ     01ah            ;End of file marker
 = 0009               tab     equ     09h             ;ASCII tab character

 = 0080               command equ     80h             ;buffer for command tail


 0000                 cseg    segment para public 'CODE'

                              assume cs:cseg,ds:data,es:data,ss:stack


 0000                 clean   proc    far             ;entry point from PC-DOS

 0000  1E                     push    ds              ;save DS:0000 for final
 0001  33 C0                  xor     ax,ax           ;return to PC-DOS
 0003  50                     push    ax
 0004  B8 —— R                mov     ax,data         ;make our data segment
 0007  8E C0                  mov     es,ax           ;addressable via ES register
 0009  E8 0097 R              call    infile          ;get path and file spec.
                                                      ;for input file
 000C  8C C0                  mov     ax,es           ;set DS=ES for remainder
 000E  8E D8                  mov     ds,ax           ;of program
 0010  73 06                  jnc     clean1          ;jump, got acceptable name
 0012  BA 00E8 R              mov     dx,offset msg4  ;missing or illegal filespec,
 0015  EB 7B 90               jmp     clean9          ;print error message and exit.

 0018  E8 00BA R      clean1: call    outfile         ;set up output filename
 001B  E8 00DF R              call    open_input      ;now try to open input file
 001E  73 06                  jnc     clean2          ;jump,opened input ok
 0020  BA 008D R              mov     dx,offset msg1  ;open of input file failed,
 0023  EB 6D 90               jmp     clean9          ;print error msg and exit.

 0026                 clean2:
 0026  E8 00EC R              call    open_output     ;try to open output file.
 0029  73 06                  jnc     clean3          ;jump,opened ok
 002B  BA 00A9 R              mov     dx,offset msg2  ;open of output file failed,
 002E  EB 62 90               jmp     clean9          ;print error message and exit.

                                                      ;files successfully opened,
 0031                 clean3:                         ;now filter the file.
 0031  E8 010B R              call    get_char        ;read 1 character from input.
 0034  24 7F                  and     al,07fh         ;strip off the high bit
 0036  3C 20                  cmp     al,20h          ;is it a control code?
 0038  73 1D                  jae     clean4          ;no,write it to new file
                                                      ;yes it is control code,
 003A  3C 1A                  cmp     al,eof          ;is it end of file marker?
 003C  74 48                  je      clean6          ;yes,jump to close files.
 003E  3C 09                  cmp     al,tab          ;is it a tab command?
 0040  74 2A                  jz      clean5          ;yes,jump to special processing.
 0042  3C 0D                  cmp     al,cr           ;if control code other than
 0044  74 08                  je      clean35         ;tab or end-of-file mark, throw
 0046  3C 0C                  cmp     al,ff           ;it away unless it is a
 0048  74 04                  je      clean35         ;form feed, carriage return,
 004A  3C 0A                  cmp     al,lf           ;or line feed.
 004C  75 E3                  jne     clean3
 004E                 clean35:                        ;If it is one of those three,
 004E  C7 06 008B R 0000      mov     column,0        ;incidentally initialize
 0054  EB 05 90               jmp     clean45         ;column count for tab processor.

 0057                 clean4:                         ;count alphanumeric chars. sent.
 0057  FF 06 008B R           inc     column

 005B                 clean45:                        ;write this character to
 005B  E8 0124 R              call    put_char        ;output file,
 005E  73 D1                  jnc     clean3          ;if CY not set, write was
                                                      ;ok so go get next char.
 0060  E8 00F9 R              call    close_input     ;if CY set, disk is full
 0063  E8 0102 R              call    close_output    ;so close files and exit
 0066  BA 00FF R              mov     dx,offset msg5  ;with error message.
 0069  EB 27 90               jmp     clean9

 006C                 clean5:                         ;process tab character
 006C  A1 008B R              mov     ax,column       ;let DX:AX=column count
 006F  99                     cwd
 0070  B9 0008                mov     cx,8            ;divide it by eight. . .
 0073  F7 F9                  idiv    cx
 0075  2B CA                  sub     cx,dx           ;remainder is in DX.
 0077  01 0E 008B R           add     column,cx       ;update column pointer.
 007B                 clean55:                        ;8 minus the remainder
 007B  51                     push    cx              ;gives us the number of
 007C  B0 20                  mov     al,20h          ;spaces to send out to
 007E  E8 0124 R              call    put_char        ;move to the next tab position
 0081  59                     pop     cx
 0082  E2 F7                  loop    clean55
 0084  EB AB                  jmp     short clean3    ;get next character

 0086                 clean6:                         ;end of file detected,
 0086  E8 0124 R              call    put_char        ;write end-of-file marker,
 0089  E8 00F9 R              call    close_input     ;close input and output
 008C  E8 0102 R              call    close_output    ;files.
 008F  BA 00C9 R              mov     dx,offset msg3  ;addr of success message,

 0092                 clean9:                         ;print message and return
 0092  B4 09                  mov     ah,9            ;control to PC-DOS
 0094  CD 21                  int     21h
 0096  CB                     ret

 0097                 clean   endp


 0097                 infile  proc    near            ;process name of input file
                                                      ;DS:SI <— addr command line
 0097  BE 0080                mov     si,offset command
                                                      ;ES:DI <— addr filespec buffer
 009A  BF 0000 R              mov     di,offset input_name
 009D  FC                     cld
 009E  AC                     lodsb                   ;any command line present?
 009F  0A C0                  or      al,al           ;return error status if not.
 00A1  74 15                  jz      infile4
 00A3                 infile1:                        ;scan over leading blanks
 00A3  AC                     lodsb                   ;to filename
 00A4  3C 0D                  cmp     al,cr           ;if we hit carriage return
 00A6  74 10                  jz      infile4         ;filename is missing.
 00A8  3C 20                  cmp     al,20h          ;is this a blank?
 00AA  74 F7                  jz      infile1         ;if so keep scanning.

 00AC                 infile2:                        ;found first char of name,
 00AC  AA                     stosb                   ;move last char. to output
                                                      ;filename buffer.
 00AD  AC                     lodsb                   ;check next character, found
 00AE  3C 0D                  cmp     al,cr           ;carriage return yet?
 00B0  74 04                  je      infile3         ;yes,exit with success code
 00B2  3C 20                  cmp     al,20h          ;is this a blank?
 00B4  75 F6                  jne     infile2         ;if not keep moving chars.

 00B6                 infile3:                        ;exit with carry =0
 00B6  F8                     clc                     ;for success flag
 00B7  C3                     ret

 00B8                 infile4:                        ;exit with carry =1
 00B8  F9                     stc                     ;for error flag
 00B9  C3                     ret
 00BA                 infile  endp

 00BA                 outfile proc    near            ;set up path and file
 00BA  FC                     cld                     ;name for output file.
 00BB  B9 0040                mov     cx,64           ;length to move
 00BE  BE 0000 R              mov     si,offset input_name   ;source addr
 00C1  BF 0040 R              mov     di,offset output_name  ;dest addr
 00C4  F3/ A4                 rep movsb               ;transfer the string
 00C6  BF 0040 R              mov     di,offset output_name
 00C9                 outfile1:                       ;scan string looking for
 00C9  8A 05                  mov     al,[di]         ;"." marking start of extension
 00CB  0A C0                  or      al,al           ;or zero byte marking name end.
 00CD  74 07                  jz      outfile2        ;if either is found,jump.
 00CF  3C 2E                  cmp     al,'.'
 00D1  74 03                  je      outfile2        ;bump string pointer, loop
 00D3  47                     inc     di              ;if neither '.' or zero found.
 00D4  EB F3                  jmp     outfile1
 00D6                 outfile2:                       ;found zero or '.',force the
                                                      ;extension of the output file
                                                      ;to '.CLN'
 00D6  BE 0086 R              mov     si,offset outfile_ext
 00D9  B9 0005                mov     cx,5
 00DC  F3/ A4                 rep movsb
 00DE  C3                     ret                     ;back to caller
 00DF                 outfile endp

 00DF                 open_input proc near            ;open input file
                                                      ;DS:DX=addr filename
 00DF  BA 0000 R              mov     dx,offset input_name
 00E2  B0 00                  mov     al,0            ;AL=0 for read only
 00E4  B4 3D                  mov     ah,3dh          ;function 3dh=open
 00E6  CD 21                  int     21h             ;handle returned in AX,
 00E8  A3 0080 R              mov     input_handle,ax ;save it for later.
 00EB  C3                     ret                     ;CY is set if error
 00EC                 open_input endp

 00EC                 open_output proc near           ;open output file
                                                      ;DS:DX=addr filename
 00EC  BA 0040 R              mov     dx,offset output_name
 00EF  B0 01                  mov     al,1            ;AL=1 for write only
 00F1  B4 3C                  mov     ah,3ch          ;function 3ch=MAKE or
 00F3  CD 21                  int     21h             ;truncate existing file
                                                      ;handle returned in AX
 00F5  A3 0082 R              mov     output_handle,ax ;save it for later.
 00F8  C3                     ret                     ;return CY=true if error
 00F9                 open_output endp

 00F9                 close_input proc near           ;close input file
 00F9  8B 1E 0080 R           mov     bx,input_handle ;BX=handle
 00FD  B4 3E                  mov     ah,3eh
 00FF  CD 21                  int     21h
 0101  C3                     ret
 0102                 close_input endp

 0102                 close_output proc near          ;close output file
 0102  8B 1E 0082 R           mov     bx,output_handle ;BX=handle
 0106  B4 3E                  mov     ah,3eh
 0108  CD 21                  int     21h
 010A  C3                     ret
 010B                 close_output endp

 010B                 get_char proc   near            ;read one character
                                                      ;from input file
 010B  8B 1E 0080 R           mov     bx,input_handle ;BX = handle
 010F  B9 0001                mov     cx,1            ;CX=length to read
                                                      ;DS:DX=addr of buffer
 0112  BA 0084 R              mov     dx,offset input_buffer
 0115  B4 3F                  mov     ah,3fh
 0117  CD 21                  int     21h
 0119  0B C0                  or      ax,ax           ;was anything read?
 011B  74 04                  jz      get_char1       ;no,end of file.
 011D  A0 0084 R              mov     al,input_buffer ;yes, return data in AL
 0120  C3                     ret
 0121                 get_char1:                      ;end of file found
 0121  B0 1A                  mov     al,eof          ;return end-of-file
 0123  C3                     ret                     ;marker in AL.
 0124                 get_char endp

 0124                 put_char proc   near            ;write one character
 0124  A2 0085 R              mov     output_buffer,al ;to the output file.
 0127  8B 1E 0082 R           mov     bx,output_handle ;BX = handle
 012B  B9 0001                mov     cx,1            ;CX=length to write
                                                      ;DS:DX=addr of data
 012E  BA 0085 R              mov     dx,offset output_buffer
 0131  B4 40                  mov     ah,40h
 0133  CD 21                  int     21h
 0135  3D 0001                cmp     ax,1            ;check bytes written
 0138  75 02                  jne     put_char1       ;if not=1 disk is full
 013A  F8                     clc                     ;otherwise return CY flag
 013B  C3                     ret                     ;0 to indicate success
 013C                 put_char1:                      ;come here if disk full
 013C  F9                     stc                     ;return CY flag=1 to
 013D  C3                     ret                     ;indicate write error
 013E                 put_char endp

 013E                 cseg    ends
```

```
0000                  data         segment para public 'DATA'

0000    40 [          input_name         db     64 dup (0)      ;buffer for input filespec
            00
               ]

0040    40 [          output_name        db     64 dup (0)      ;buffer for output filespec
            00
               ]


0080  0000            input_handle       dw     0               ;token returned by PCDOS
0082  0000            output_handle      dw     0               ;token returned by PCDOS

0084  00              input_buffer       db     0               ;character read from input
                                                                ;file comes here.
0085  00              output_buffer      db     0               ;character is written to
                                                                ;output file from here.

0086  2E 43 4C 4E 00  outfile_ext        db     '.CLN',0        ;extension for filtered file

008B  0000            column             dw     0               ;column count for tab processing

008D  0D 0A           msg1               db     cr,lf
008F  43 61 6E 6E 6F 74                  db     'Cannot find input file.'
      20 66 69 6E 64 20
      69 6E 70 75 74 20
      66 69 6C 65 2E
00A6  0D 0A 24                           db     cr,lf,'$'

00A9  0D 0A           msg2               db     cr,lf
00AB  46 61 69 6C 65 64                  db     'Failed to open output file.'
      20 74 6F 20 6F 70
      65 6E 20 6F 75 74
      70 75 74 20 66 69
      6C 65 2E
00C6  0D 0A 24                           db     cr,lf,'$'

00C9  0D 0A           msg3               db     cr,lf
00CB  46 69 6C 65 20 70                  db     'File processing completed.'
      72 6F 63 65 73 73
      69 6E 67 20 63 6F
      6D 70 6C 65 74 65
      64 2E
00E5  0D 0A 24                           db     cr,lf,'$'

00E8  0D 0A           msg4               db     cr,lf
00EA  4D 69 73 73 69 6E                  db     'Missing filename.'
      67 20 66 69 6C 65
      20 6E 61 6D 65 2E
00FC  0D 0A 24                           db     cr,lf,'$'

00FF  0D 0A           msg5               db     cr,lf
0101  44 69 73 6B 20 69                  db     'Disk is full.'
      73 20 66 75 6C 6C
      2E
010E  0D 0A 24                           db     cr,lf,'$'

0111                  data         ends


0000                  stack        segment para stack 'STACK'
0000    40 [                       db             64 dup (?)
            ??
               ]

0040                  stack        ends

                                   end            clean
```

Segments and groups:

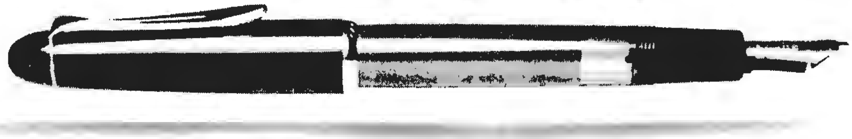
Name	Size	align	combine	class
CSEG	013E	PARA	PUBLIC	'CODE'
DATA	0111	PARA	PUBLIC	'DATA'
STACK	0040	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr	
CLEAN	F PROC	0000	CSEG	Length =0097
CLEAN1	L NEAR	0018	CSEG	
CLEAN2	L NEAR	0026	CSEG	
CLEAN3	L NEAR	0031	CSEG	
CLEAN35	L NEAR	004E	CSEG	
CLEAN4	L NEAR	0057	CSEG	
CLEAN45	L NEAR	005B	CSEG	
CLEAN5	L NEAR	006C	CSEG	
CLEAN55	L NEAR	007B	CSEG	
CLEAN6	L NEAR	0086	CSEG	
CLEAN9	L NEAR	0092	CSEG	
CLOSE_INPUT	N PROC	00F9	CSEG	Length =0009
CLOSE_OUTPUT	N PROC	0102	CSEG	Length =0009
COLUMN	L WORD	008B	DATA	
COMMAND	Number	0080		
CR	Number	000D		
EOF	Number	001A		
FF	Number	000C		
GET_CHAR	N PROC	010B	CSEG	Length =0019
GET_CHAR1	L NEAR	0121	CSEG	
INFILE	N PROC	0097	CSEG	Length =0023
INFILE1	L NEAR	00A3	CSEG	
INFILE2	L NEAR	00AC	CSEG	
INFILE3	L NEAR	00B6	CSEG	
INFILE4	L NEAR	00B8	CSEG	
INPUT_BUFFER	L BYTE	0084	DATA	
INPUT_HANDLE	L WORD	0080	DATA	
INPUT_NAME	L BYTE	0000	DATA	Length =0040
LF	Number	000A		
MSG1	L BYTE	008D	DATA	
MSG2	L BYTE	00A9	DATA	
MSG3	L BYTE	00C9	DATA	
MSG4	L BYTE	00E8	DATA	
MSG5	L BYTE	00FF	DATA	
OPEN_INPUT	N PROC	00DF	CSEG	Length =000D
OPEN_OUTPUT	N PROC	00EC	CSEG	Length =000D
OUTFILE	N PROC	00BA	CSEG	Length =0025
OUTFILE1	L NEAR	00C9	CSEG	
OUTFILE2	L NEAR	00D6	CSEG	
OUTFILE_EXT	L BYTE	0086	DATA	
OUTPUT_BUFFER	L BYTE	0085	DATA	
OUTPUT_HANDLE	L WORD	0082	DATA	
OUTPUT_NAME	L BYTE	0040	DATA	Length =0040
PUT_CHAR	N PROC	0124	CSEG	Length =001A
PUT_CHAR1	L NEAR	013C	CSEG	
TAB	Number	0009		

Warning Severe
Errors Errors
0 0

BUSINESS LETTERS TO GO.*



*OR, HOW TO WRITE EFFECTIVE BUSINESS LETTERS WITHOUT REALLY WRITING.

Often, when it comes to business letter writing, it's difficult to find the time. Or even the right words.

Add the routine business letters you should write during each busy day, and writing becomes an endless chore.

Even if you use a personal computer.

So you end up *not* writing all the letters you should. Or working overtime just to catch up.

Now there's help.

BUSINESS LETTERS AT YOUR FINGERTIPS.

With The Business Letter Library™, you get hundreds of concise, unique business letters, on disk. Superbly written by a leading business communications expert.

They're ready to go. And ready to use with your personal computer and almost any popular word processing software.

All *you* have to do is select the letter you need; customize, print, and mail. To clients, customers, business associates, vendors; *anyone* you do business with.

You get letters to start a business deal. Set up a meeting. Make a point. Close a sale. And *more*.

BUSINESS LETTERS WHEN YOU NEED THEM. FAST.

The LetterSearch™ feature of The Business Letter Library helps you quickly locate the type of business letter you need.

For routine correspondence, you can make quick changes to any letter from The Business Letter Library.

Or use the on-screen KeyStyle™ mode to write a custom, highly personalized version of each letter.

Either way, The Business Letter Library can help you write effective business letters to fit almost any business situation.

YOU'RE DONE!

By the time you've read this ad, you could've written a great business letter. It's really that easy.

And it's really that good: Whether you need to write many routine letters, or that single, very important one, you'll get *results*.

So turn business letter writing into a quick, easy, productive part of your day with The Business Letter Library.

For a full demonstration of The Business Letter Library, see your Dealer. Or call us at **(714) 642-1827.**

The Business Letter Library™

The Business Letter Library can be used with the IBM Personal Computer, Apple II or IIe, DEC Rainbow, Commodore 64 and many other microcomputers. Compatible with most popular word processing software packages.

Delta Point®

711 West 17th Street, Unit B10, Costa Mesa, CA 92627
(714) 642-1827

Look for future products in the Letter Library Series to meet your business, professional, and your social letter-writing needs.

softalk presents the bestsellers

Why should *Softalk* be different? It's the new year, and we'll be turning over a few new leaves of our own. One or two may even make sense.

We're going to stop searching for new metaphors to explain *1-2-3*'s dominance in the market. No more lines like: "*1-2-3* is so strong that, if it were an onion, a whole bottle of Listerine and a whole package of Certs couldn't freshen your breath."

No more mystical approaches to *1-2-3*, either. You won't be reading lines like this in Bestsellers: "*1-2-3* is so good, if there weren't an IBM PC to run it on, somebody would have to invent one."

You won't be reading any superlatives about *1-2-3* progenitor Mitch Kapor, either, such as: "Mitch Kapor has mastered the alchemist's art of turning bytes into gold."

Missing from these pages will be non sequiturs playing off the publisher's name: "It's not surprising that Lotus's stock remains high. They've assumed the correct position."

We'll also restrain ourselves when writing about *WordStar*.

No more "If *WordStar* were Methuselah's age, we'd still be in 1982."

It's time to stop beating the dead horse about whether *WordStar* can hold its lead in word processing. "*WordStar* clings to the lead in word processing with the tenacity of a spinster seeing the last eligible bachelor in town boarding the bus."

Let's mute the growing paeans of admiration for Softword Systems, publisher of *MultiMate*: "The publisher of *MultiMate* is smart enough to learn by example; after all, Xerox makes lots of money off copies."

And we'll forego belaboring the obvious lesson of *MultiMate*'s success: "The way to sell successfully to the Fortune 500 is to have a product so much like what they already know that it's nonthreatening. Different is a pejorative description at the corporate level."

This is the year that we'll stop calling everything except word processors and spreadsheets a database management system. *Condor* and *dBase II* are database management systems. Most of the rest are not.

No sense in marveling over the widespread acceptance of *dBase II* . . . "a program certainly as easy to understand as Heisenberg's uncertainty principle."

It's also time to stop speculating on what all those corporate types are doing with their PCs. The sale of communications packages tells the whole story. "*Crosstalk* may describe the tone of the corporate middle manager while trying to communicate with the company mainframe."

With *1-2-3* chewing up the competition, it's time to get realistic about spreadsheets:

"*Multiplan* was written in 1981, tested in 1982, sold in 1983, and obsolete in 1984. . .a true microcosm of the microcomputer industry."

"*VisiCalc* had stopped being software and become a concept. Watch out if 'What if' ever turns into 'So what.'"

"*SuperCalc* is trying to ratify the concept of reincarnation, but few people are comforted by thinking they'll return to Earth as a fruit fly."

This is the year to stop bemoaning the indifference of PC owners to entertainment software. "No wonder adventures don't sell. For most middle managers, solving the puzzle of obtaining a key to the executive washroom is challenging enough."

"*Pac-Man* is only a symbol. The maze of office politics is real. Why settle for the ephemeral when you can devote yourself to the real thing?"

Let's stop marveling at the plethora of software to teach system and software usage to new PC buyers and put the obvious conclusion out in the open air:

"Either the IBM Personal Computer is more complicated to run than

any micro that preceded it, or its buyers, as a class, are exceedingly backward. The alternative hypothesis is that PC buyers are too lazy to learn, so they require being taught."

It's time to recognize finally the real value of Microsoft's *Flight Simulator*:

"Bruce Artwick's *Flight Simulator* is true art, which speaks well of the PC market that keeps buying it. It's just too bad that so many people have to claim they're readying for their pilot's exam to justify the purchase."

This will also be the year to keep our cynicism about newer advances in check.

"It's not clear whether hard disk refers to the condition of the medium or the ease with which it's kept operating."

"The concept of windows on a microcomputer has all the appeal of a McDonald's on the Ginza."

"Most IBM PC owners won't do windows at home or at the office, and they won't do windows on their PC either, until the concept gets considerably refined."

"IBM says they'll triple the manufacturing of PCs this year. What Compaq wants to know is whether IBM will get around to shipping them as well."

"If IBM really does ship thrice as much product, will other manufacturers continue to get away with cloning around?"

"Since IBM owners don't play arcade games, how will they learn to aim the keyboard of the PC*jr*?" ▲

the top thirty

This Month	Last Month	Index	
1.	1.	331.08	**1-2-3**, Mitch Kapor and Jonathan Sachs; Lotus Development
2.	2.	97.34	**WordStar**; MicroPro
3.	4.	86.64	**MultiMate**; Softword Systems
4.	10.	84.51	**Microsoft Flight Simulator**, Bruce Artwick; SubLogic
5.	3.	59.37	**dBase II**, Wayne Ratliff; Ashton-Tate
6.	6.	50.81	**PFS:File**, John Page and D.D. Roberts; Software Publishing Corporation
7.	24.	41.72	**MasterType**, Lightning Software/Bruce Zweig; Scarborough Systems
8.	16.	39.58	**Macro Assembler**, Microsoft; IBM
9.	5.	37.97	**Crosstalk**; Microstuf
10.	20.	33.69	**Home Accountant Plus**, Mike Farmer, Bob Schoenburg, Larry Grodin, and Steve Pollack; Continental Software
11.	14.	30.48	**Basic Compiler**, Microsoft; IBM
12.	12.	28.88	**Norton Utilities**, Peter Norton; Peter Norton Inc.
13.	8.	27.81	**Multiplan**, Microsoft; IBM
14.	26.	26.74	**Typing Tutor**, Michael Sierchio (Dick Ainsworth and Al Baker); IBM (Microsoft)
15.	8.	26.20	**PFS:Write**, Sam Edwards, Brad Crain, and Ed Mitchell; Software Publishing Corporation
16.	21.	23.53	**The Instructor**, Jo-L Hendrickson; Individual Software
17.	28.	21.92	**Zork I**; Infocom
18.	7.	21.39	**VisiCalc**, Software Arts/Dan Bricklin and Robert Frankston; VisiCorp, IBM
19.	—	20.32	**Zork III**; Infocom
	17.	20.32	**PC Tutor**, Lora Meise and Rick Lane; Comprehensive Software Support
21.	23.	16.58	**PFS:Report**, John Page; Software Publishing Corporation
	12.	16.58	**WordPerfect**, Alan Ashton and Bruce Bastian; Satellite Software International
23.	19.	13.90	**Cdex Training for the IBM PC**, Rohit Patel; Cdex Corporation
24.	14.	13.37	**SuperCalc2**; Sorcim
	—	13.37	**Zork II**; Infocom
	—	13.37	**EasyWriter II**, Basic Software Group; Information Unlimited Software
27.	22.	12.83	**Volkswriter**, Camilo Wilson; Lifetree
28.	—	11.76	**Word**; Microsoft
	26.	11.76	**PFS:Graph**, Bessie Chin and Stephen Hill; Software Publishing Corporation
30.	25.	10.69	**ProKey**, David Rose; RoseSoft

IBM-franchised retail stores representing approximately 4.93 percent of all sales of IBM and IBM-related products volunteered to participate in the poll.

Respondents were contacted early in December to ascertain their sales for the month of November.

The only criterion for inclusion on the list was the number of units sold; such other criteria as quality of product, profitability to the computer store, and personal preference of the individual respondents were not considered.

Respondents in December respresented every geographical area of the continental United States.

Results of the responses were tabulated using a formula that resulted in the index number to the left of the program name in the Top Thirty listing. The index number is an arbitrary measure of relative strength of the programs listed. Index numbers are correlative only to the month in which they are printed; readers cannot assume that an index rating of 50 in one month represents equivalent sales to an index number of 50 in another month.

Probability of statistical error is plus or minus 3.88 percent, which translates roughly into the theoretical possibility of a change of 4.93 points, plus or minus, in any index number.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 673 NO. HOLLYWOOD, CA

POSTAGE WILL BE PAID BY ADDRESSEE

softalk

for the IBM Personal Computer

P.O. Box 60
North Hollywood, California 91603


$95
TEXTRA
Al Savage '83
Textra™
word processing
The shortest distance between your thoughts and the printed word.
Your thoughts come faster than your words. Capturing them smoothly and quickly is the strength of Textra, and the secret behind its success.
Textra blends advanced features with a simplified design that makes it easier to write with your personal computer. And that's what word processing is all about.
Start with the 'on-line' tutorial, which has been called everything from elegant to excellent. It's the finest introduction to a software program ever filmed. You'll be ready to write in record time.
And Textra will be ready for you. Its highly responsive full screen editor helps you put your thoughts on the screen swiftly. Automatic reformatting and on-screen bolding and underlining show you what your printed text will look like at all times. You can even preview your pages *before* they're printed, so you only have to print them once!
Whether you're searching for your first word processor, or feel miserable with the one you have now, take a close look at Textra. Ask your dealer for a demonstration today.
It's guaranteed to open your eye.
A² Software
Textra runs on the IBM PC and compatibles, and requires PCDOS (any version), 128K, and either monitor. Textra Jr. ($39.95) requires 64K.
Ann Arbor Software
407 N. Main, Ann Arbor, MI 48104 Phone (313) 769-9088
© 1983, Ann Arbor Software. All rights reserved.

Many happy returns.

If filing a return always leaves you frazzled by forms, rattled by receipts and numbed-out by numbers, HowardSoft can help you lick this taxing situation. With HowardSoft Tax Preparer and your IBM-PC or Apple Computer, you can take the tax break you deserve.

Using HowardSoft, your tax form is right on the screen where you need it. You enter raw information just once as HowardSoft calculates quickly and accurately, makes changes automatically, then delivers error-free print-outs ready to sign and drop in the mail.

Plus, you can keep records and make tax projections year-round. HowardSoft gives you all the features of the high-priced packages at a fraction of the cost. Clear instructions, the most-used forms and schedules, and inexpensive annual updates for easy filing year after year after year.

So, why not give HowardSoft a try? Visit the computer store nearest you for a demonstration of the top-selling tax package in the country. You can count on returning happy.

Tax Preparer by HowardSoft.™
The #1 selling tax software.
8008 Girard Avenue, Suite 310, La Jolla, CA 92037 • (619) 454-0121